

MTPS: A MULTI-TIME-FRAME TECHNICAL PATTERN AND STRATEGY BACKTESTING SYSTEM

by

DIWEN XU

A Thesis Submitted to
The Hong Kong University of Science and Technology
in Partial Fulfillment of the Requirements for
the Degree of Master of Philosophy
in Computer Science and Engineering

July 2018, Hong Kong

Copyright © by Diwen Xu 2018

Authorization

I hereby declare that I am the sole author of the thesis.

I authorize the Hong Kong University of Science and Technology to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize the Hong Kong University of Science and Technology to reproduce the thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

A handwritten signature in black ink, appearing to be 'Diwen Xu', written over a horizontal line.

DIWEN XU

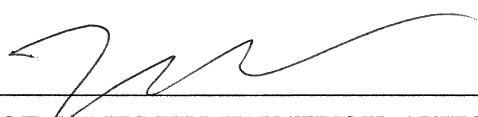
27 July 2018

MTPS: A MULTI-TIME-FRAME TECHNICAL PATTERN AND STRATEGY BACKTESTING SYSTEM

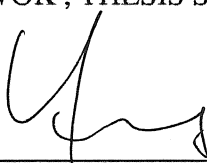
by

DIWEN XU

This is to certify that I have examined the above M.Phil. thesis
and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by
the thesis examination committee have been made.



PROF. JAMES TIN-YAU KWOK , THESIS SUPERVISOR



PROF. DIT-YAN YEUNG, ACTING HEAD OF DEPARTMENT

Department of Computer Science and Engineering

27 July 2018

TABLE OF CONTENTS

Title Page	i
Authorization Page	ii
Signature Page	iii
Table of Contents	iv
List of Figures	vii
List of Tables	ix
Acknowledgments	x
Abstract	xi
Chapter 1 Introduction	1
1.1 Motivation	1
1.2 Contribution	3
1.3 Thesis organization	4
Chapter 2 Related Work	5
2.1 Algorithmic Trading Strategies	5
2.2 Strategy Backtesting System	7
2.2.1 Vector-based Simulation and Event-driven Simulation	7
2.2.2 Data storage design and Database selection	9
2.2.3 Sequence Matching over Event Streams	11
2.2.4 Strategy describing Interface	12

Chapter 3	Technical Analysis Strategies	14
3.1	Technical Indicators	14
3.1.1	Single-Bar Indicators	15
3.1.2	Price-Level Indicator	17
3.2	Technical Patterns	20
3.3	Multi-Time-Frame Technical Analysis	22
3.4	Strategy and performance evaluation	23
Chapter 4	MTPS Strategy Description System and Graphical User Interface	25
4.1	Event Layer	29
4.2	Pattern Layer	32
4.3	Modeling Multi-time-frame Pattern	35
4.4	Strategy Layer and Simulation-Page	36
4.5	Expressive Power and Limitations of MTPS Pattern Description	38
Chapter 5	MTPS Design and Implementation	41
5.1	MTPS Pattern Analyzer	43
5.1.1	Multi-time-frame Candle Indexing	44
5.1.2	Event Detection	44
5.1.3	Pattern Matching	45
5.1.4	Time-frame Alignment	49
5.2	MTPS Market Data Pre-processing	50
5.3	Technical Indicators Implemented	54
5.4	MTPS Trading Components	57
5.5	Event-Driven Architecture of MTPS	60
Chapter 6	Experimental Evaluation	62
6.1	Experimental Setup	62
6.2	Experiment Results on "Day-50 break pull-back"	63
6.3	Experiments Results on "Hourly RSI Divergence"	67
6.4	Experiment Results on "Down Trend Exhausted"	69

Chapter 7 Conclusion and Future Works	75
7.1 Conclusion	75
7.2 Future Works	76
References	78

LIST OF FIGURES

2.1	components of real-time backtesting system	8
2.2	Definition of candlestick chart	10
3.1	React and Breakout of Support	19
3.2	example of Support levels	19
3.3	example of Support Turn Resistance	21
4.1	strategy description page of NinjaTrader	26
4.2	Overview of MTPS GUI flow	28
4.3	search bar event structure	31
4.4	search bar event GUI	31
4.5	line-page GUI	31
4.6	search-line event GUI	32
4.7	pattern layer structure	34
4.8	pattern 60rsidiv	35
4.9	strategy-page layout	37
4.10	MTPS strategy-page GUI	37
4.11	Simulation Page GUI	38
4.12	An example of MTPS pattern as NFA	39
5.1	MTPS architecture	42
5.2	MTPS work-flow	43
5.3	time-frame alignment of Pattern P	51
5.4	Performance Report GUI	59
5.5	EDA in MTPS	61
6.1	Day-50 break pull back	64
6.2	profit of "Day-50 break pull-back" vs. "Buy and hold" on Crude Oil	66
6.3	profit of "Day-50 break pull-back" vs. "Buy and hold" on Gold	66
6.4	profit of "Day-50 break pull-back" vs. "Buy and hold" on NASDAQ	66
6.5	Typical "Hourly RSI Divergence"	68

6.6	profit of "Hourly RSI Divergence" vs. "Buy and hold" on Crude Oil	70
6.7	profit of "Hourly RSI Divergence" vs. "Buy and hold" on Gold	70
6.8	profit of "Hourly RSI Divergence" vs. "Buy and hold" on NASDAQ	70
6.9	example of "Down Trend Exhausted"	73
6.10	Performance of Strategy "Short Trend Exhausted"	74
6.11	profit of "Short Trend Exhausted" vs. "Buy and hold" on Crude Oil	74

LIST OF TABLES

2.1	Features of Strategy Designing Interface	12
6.1	Parameter Sets of Day-50 break pull-back	65
6.2	Performance of Strategy "Day-50 break pull-back"	65
6.3	Parameter Sets of Hourly RSI Divergence	69
6.4	Performance of Strategy "Hourly RSI Divergence"	69
6.5	Parameter Sets of "Short Trend Exhausted"	73

ACKNOWLEDGMENTS

It is such a privilege of mine that I can have Prof. James Kwok as my thesis supervisor. I would like express my sincere gratitude and appreciation to him for his kind and precious support and guidance during my whole Mphil study in HKUST. Without his help and patience, it's not possible for me to complete this thesis and my academic study.

I would also like to deeply thank my thesis committee members, Prof. Gray Chan and Dr. Wilfred Ng for their generous and inspiring comments on my thesis to improve my works.

In addition, I would like to give my special thanks to Mr. Jansen Leung who is a senior technical traders with more 20 years future trading experience. His plentiful knowledge and suggestions play very important roles on inspiring the idea of this thesis. Mr. Leung also serves as the first user of the software system presented in this thesis.

Last but not least, I would like to thank my parents and girl friend for their support and encouragement.

MTPS: A MULTI-TIME-FRAME TECHNICAL PATTERN AND STRATEGY BACKTESTING SYSTEM

by

DIWEN XU

Department of Computer Science and Engineering

The Hong Kong University of Science and Technology

ABSTRACT

Strategy backtesting which is a process to take some user-defined trading rules and market information as input, simulate behaviors of market participants under these trading rules and finally report the strategy performance. Strategy backtesting systems are the most important applications of algorithmic trading since it provide individual or professional traders flexibility to design and test their customized ideas. Technical analysis is the most popular way for financial instruments traders to predict future price movement by researching historical data records. However, technical analyses highly rely on a various of technical indicators, patterns and personal charting heuristics, which makes it very difficult to be transformed into algorithmic trading strategies. Most technical traders especially for individual traders, on the other hand, are lack of knowledge about computer programming, and find it hard for them to script their strategies by coding. Therefore, a strategy backtesting system allowing users to describe complex and advanced technical strategies via pure graphical user interface (GUI) is in very highly demand.

In this thesis, we design and implement a multi-time-frame technical pattern and strategy back-testing system, that is MTPS, to address the problem of describing complex technical patterns on GUI and applying these patterns in algorithmic trading strategies. By observing the cases that a large amount of profitable technical patterns are composed of sequences of characteristic candle bars, we introduce a hierarchical MTPS strategy description system that promote users to describe their technical patterns by describing sequences of technical events. This strategy description system endow users who are not knowledgeable about computer programming a much higher flexibility to include complicated and advanced patterns in their algorithmic strategies. In addition, we also provide users interfaces in MTPS to include technical indicators from multiple time-frames in one strategy which is a unique feature that most of the state-of-the-art strategy backtesting systems can not achieve. Besides, we present the complete architecture of MTPS including the implementation details of all the software components and how they are organized in an event-driven design pattern. Moreover, we propose an efficient pattern matching algorithm for MTPS to minimize the delay between market data arrival and order placement. Last but not least, experiments are conducted showing that MTPS do capable to describe 3 complex and advanced technical patterns and backtesting strategies based on these 3 patterns to generate profit surpassing the "buy-and-hold" strategies.

CHAPTER 1

INTRODUCTION

1.1 Motivation

Compared with human beings, computer programs can proceed tremendous data in very short time and won't be affected by emotion, so they are naturally outstanding traders. Times of algorithmic trading has already come. It is reported that more than 80% of trades of financial instruments are made by computers [49] at year 2012. Algorithmic trading strategies are essential components of the computer based trading, which are sets of trading rules that are executable on computing devices to decide trading operations. Nowadays, many hedge funds accumulate large amount of profit by taking use of algorithmic trading strategies. Bloomberg News once stated that some top algorithmic trading strategies ever achieved 47% annual return rate [44].

A large amount of algorithmic trading strategies are developed over 70 years. Most of the algorithmic trading strategies can be divided into two categories based on their trading principles. One category of algorithmic trading strategies focus on seeking tiny but frequent arbitraging opportunities or making profit by providing liquidity between buy side and sell side. High Frequency Trading(HFT) is the most famous representative of this category. HFT strategies such as statistical arbitrage, spread capturing, quote matching, can generate stable profit with a much lower risk compared with other kinds of strategies [17]. However, there are also significant limitations and controversy of HFT strategies. All HFT strategies are highly relied on sophisticated high-speed computer hardwares and some HFT strategies even try to unfairly exploit others' order information. A further research by Kearns stressed that, in many empirical cases, the high-liquidity assumption and low-risk assumption of HFT even don't hold[30].

The other main category of algorithmic trading strategies try to predict the future prices of financial instruments. To achieve this goal, many algorithmic trading strategies based on traditional fundamental analysis and technical analysis as well as newly quantitative analysis are designed

and implemented. Fundamental analyses are easily to be misled by price manipulation and only work well in long-term investment. Quantitative analyses, however, requires advanced mathematics knowledge or machine learning algorithms, reliable data collection and powerful computation resources, which makes them not suitable for individual investors or small organizations. On the other hand, strategies based on technical analyses show great advantages over other kinds of strategies since they are more human-understandable, easier to implement and suitable for both short-term and long-term investments. Lento also showed that technical trading rules do have profitability in theoretical manner[36]. Therefore, algorithmic trading strategies performing advanced technical analyses recently aroused significant interest in industry.

Most traders, before performing technical trading strategies, would like to simulate the behaviors of their strategies on historical market data to verify their ideas. Such processes of strategy simulation are called backtesting, and many strategy backtesting systems are developed to help traders to backtest their technical strategies automatically. The most important feature of strategy backtesting systems is the strategy description interfaces promoting users to describe their logics of strategies and transforming these logics into algorithmic strategies. However, the description interfaces of most state-of-the-art backtesting systems have serious limitations. On one hand, many backtesting systems provide programming interfaces allowing users to code their logics with some specific programming languages. While, many traders especially for individual traders and traditional traders have little knowledge about computer programming and can not code their strategies correctly. Some backtesting systems even ask users to script strategies with their dedicated programming language which makes the code of strategies not reusable in other platforms. On the other hand, graphical user interfaces (GUI) for strategy description also offered by a lot of backtesting systems, but all these GUI can represent very simple logics of strategies which restricts the flexibility of traders to quantify their trading ideas. Many GUI only allow to include one kind of technical indicator in one strategy, and some other GUI just permit users to configure parameters of indicators of system-predefined strategies. These kinds of GUI obviously can not fulfill the requirements of traders, and therefore, a technical strategy backtesting systems that allow user to build strategy with pure GUI to describe complex and advanced technical patterns and strategies are in highly demand.

In addition, another important limitation of all the existing strategy backtesting systems we've investigated is that users are not able to describe any multi-time-frame technical patterns. Multi-time-frame technical patterns refer to technical patterns including technical indicators from more than one time-frames. Since the main target of technical analysis is to capturing trends in market movements, multi-time-frame technical patterns can overall consider the impacts of long-term and short-term market sentiments and generate trading signals more accurately. The power of multi-time-frame technical analysis has already been proved by many traders and a lot of multi-time-frame technical patterns[32][25][1] are developed. However, all the commercial backtesting systems we investigated just treat time-frame as strategy level variables, which means for one technical pattern or strategy, all the technical indicators have to be in one single time-frame.

There are still not many works conducted to address the above limitations of existing strategy backtesting systems so far. Therefore, our work MTPS aims to design and implement a multi-time-frame technical pattern and strategy backtesting framework, allowing users to build complex, multi-time-frame, customized technical strategies with pure GUI.

1.2 Contribution

The main contributions of MTPS is mainly endowed by the following aspects:

1. MTPS introduce a unique 3-layer hierarchical technical strategy description system. This strategy description system allow users to describe most complex and advanced technical patterns in strategies on pure graphical user interface (GUI) compared with all the state-of-the-art GUI based strategy description systems.
2. We implement a rich set of most commonly used technical indicators. We also produce GUI which enables users to include indicators calculated on candle charts of different time-frames which forms a very simple way to describe multi-time-frame technical strategies.
3. We present a completed architecture of strategy backtesting systems. The design and implementation of all the software components including "market data preprocessing" component, "pattern analyzer" component, "trading engine", "order management system" and "strategy

performance analyzer” are illustrated. And we also show that how these components are organized in an event-driven design pattern.

4. We transform the problem of detecting technical patterns on market data streams into the problem of matching technical event sequences on event streams. We solve the transformed problem by proposing a ”fast pattern matching” algorithm and then analyze its time complexity and show how this algorithm can significantly reduce delay in real-time trading.

In this thesis, we experiment and verify all the functionalities of MTPS by implementing and backtesting several typical technical strategies on historical price data of Crude Oil, Gold and NASDAQ index futures traded on NYMEX and COMEX.

1.3 Thesis organization

The remainder of my thesis is organized as follows. Chapter 2 reviews some related works on algorithmic strategies and the state-of-the-art strategy backtesting systems. In chapter 3, we introduce background knowledge about technical analysis and strategy evaluation involved in MTPS. In chapter 4, we propose the 3-layer hierarchical technical strategy description system and its corresponding GUI. we also discuss about the reason behind such hierarchical strategy description system and suggest its advantages over all the state-of-the-art strategy backtesting systems. Chapter 5 presents the design and implementation of MTPS, including the architecture, technical analysis component and pattern matching algorithm, market data preprocessing procedures and trading related components. In chapter 6, experiments to verify and evaluate the power of MTPS are conducted. In the experiments, MTPS are employed to design and backtest three technical pattern based strategies and report their performance. Chapter 7 summarize the thesis and outline future research directions.

CHAPTER 2

RELATED WORK

In this chapter, we first summarize some related works of algorithmic trading strategies and then detailedly review several designing and implementation considerations on some state-of-the-art research or commercial strategy backtesting systems.

2.1 Algorithmic Trading Strategies

Algorithmic trading is a process to take use of computer algorithms to handle price movements of financial instruments and send trading orders to brokers or exchanges. Algorithmic trading shows significant advantages over manually trading. Computer softwares and networks allow traders to place orders more instantly and accurately and execute orders at best possible prices. Algorithms can also help traders to analyze and forecast market price movement by historical price, volume, charting with mathematical models. Algorithmic trading not only generate profit for the algorithm owners but also benefits the whole market by increase liquidity. Algorithmic trading strategies can be mainly divided into two major branches by methodologies and targets as follows.

1. **Arbitraging and Order Execution Optimization:** Arbitraging and Order execution optimization are trading methods for traders to take tiny but low risk profits from other market participants. Arbitrage is a process to simultaneously purchase and sale of assets to take profit from differences in the prices. The simplest arbitrage is the Market Neutral Arbitrage, which try to buy and sell securities or derivatives that are naturally correlated such as stocks and futures about the same company or index. There are more complex arbitrage strategies such as Statistical Arbitrage, try to use mathematical models to find correlations between different instruments, and then buy and sell them. In 2008, Avellaneda first summarized two famous Statistical Arbitrage algorithms, Principal Components Analysis based arbitraging

and ETF arbitraging [16]. Other mathematical models e.g. Cointegration [20] and Flexible Least Squares [41] show very good profitability. Order Execution Optimization are trading methods to split large orders to relatively small orders. The principle behind these methods are to make orders to be executed at best possible price and minimize transaction cost. One of the most typical order execution optimization strategies is the Volume Weighted Average Price (VWAP) strategy. VWAP strategies are the strategies that orders are executed close to the market VWAP. Biakowski proposed two Dynamic Volume Trading Model at 2006 to implement VWAP strategies [18]. Other order execution optimization strategies include Time Weighted Average Price strategies, Stochastic Price Recovery [28], Limit Order Book Model [12] are developed and widely applied. Nevmyvaka also proposed a Reinforcement Learning based algorithm to optimize order execution [43].

2. **Forecasting future price:** Forecasting the market is an academic topic which has been investigated by various researchers and an important goal for all the market participants. Market prediction strategies either take advantages of traditional trading techniques such as fundamental analysis and technical analysis, or apply modern mathematical models. Many researches and application use text mining to learn market sentiments or fundamentals about companies from news or posts on social medias to predict stock prices [42, ?, 26]. Another major branch of methods to forecast market price is to apply time series mining algorithms to recognize patterns or signals indicating bullish or bearish price movements. Artificial Neural Network algorithms are powerful tools to do the market prediction [33, 35, 29, 22], and Support Vector Machine algorithms are also extensively applied on market forecasting [51, 37, 39]. Last but not least, technical analysis skills and knowledge are also implemented in algorithmic trading. For instance, Genetic algorithms are quite often applied to extract more complex technical trading rules [40, 13, 45].

In this thesis, we will design and implement an algorithmic trading system that based on technical analysis market forecasting. Technical analysis algorithms are computational simple and highly human-understandable, that is perfectly adequate for traditional fund managers or individual traders. Instead of hard-coding any technical strategies, our system, MTPS, offer users candle bars, technical indicators, and other building blocks to build their own strategies and run the user-

defined strategies on historical market data giving users the performances. In next section, we will propose a brief review the strengths and weaknesses of the state-of-the-art technical strategy backtesting system.

2.2 Strategy Backtesting System

Strategy Backtesting is a process to simulate behavior of strategies on historical market data and present their performance for traders to analyze the profitability and risk of strategies. A basic strategy backtesting system should consist of at least the functionalities that allow users to access historical market data, design and describe logic of strategies, detect trading signals based on the change of market price and volume, simulate behaviors of human traders when there are trading signals, evaluate the performances of strategies. More sophisticated features of backtesting systems includes risk control and money management system, build-in tools such as indicators or AI models to recognize market signals, strategy configuration management and parameters optimization. Real-time strategy backtesting mainly refers to backtesting on a real-time data stream. Real-time strategy backtesting systems are easily to be extended to live automated trading systems by connecting with bank account and trading brokers. Figure 2.1 shows the necessary software components of a real-time strategy backtesting system. In the subsections below, we will give brief reviews of related papers and successful commercial strategy backtesting systems on several essential components.

2.2.1 Vector-based Simulation and Event-driven Simulation

Strategy simulation is the core functionality of a backtesting system, which is to take time series market price data and other market information as input and output sequences of buy and sell trading decisions. Backtesting systems can be mainly divided into two categories by the implementation of strategy simulation, that are vector-based simulation systems and event-based simulation systems.

The terminology "vector-based" refers to a simulation process that all the market data such as open, high, low, close prices of candlestick bars, are pre-generated and stored in vectors, and

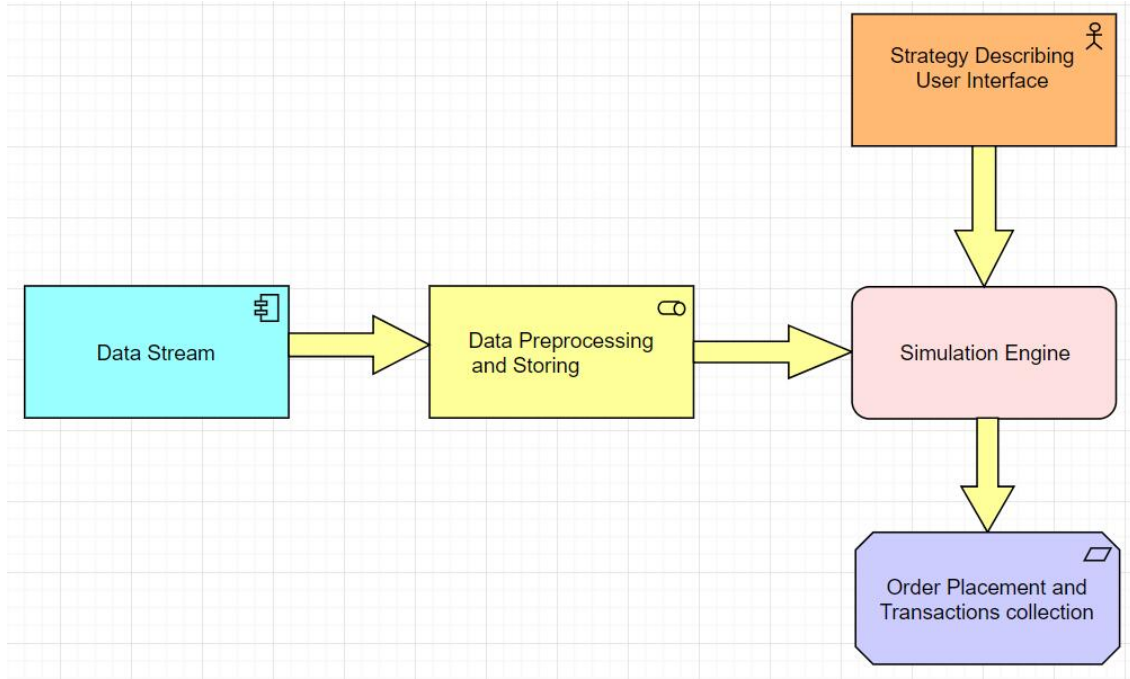


Figure 2.1: components of real-time backtesting system

market decisions are extracted from these vectors. The main advantages of vector-based simulation is time-efficient in execution and easy to implement. However, its defects are also fatal due to the so-called "Look-ahead Bias". The bias refers to use the data that is not available at certain time-stamp in live-trading to do strategy backtest. For example, if a strategy tries to buy a stock when the price cross above Day-10(last 10 days' moving average), vector-based simulation may directly compare the price with the corresponding pre-calculated Day-10 in the vector. But in live-trading, the Day-10 can only be calculated after day market closed, which is not possible for a user to place any orders. Therefore, suffering from the "Look-ahead Bias", vector-based simulation is mainly used for research/experiment purposes, not suitable for commercial or investment usages.

On the other hand, most of the commercial or practical strategy backtesting systems adopt the event-driven simulation. Event-driven simulation is derived from event-driven programming pattern in software engineering. In this pattern, the softwares are running permanent loops awaiting for some events which may be any external information such as user inputs, network requests or messages from other programs, and will response to or handle these events. Event-driven program-

ming is very commonly used in software implementation especially for GUI implementation due to its advantages of robustness and conciseness [24]. In the event-driven simulation, backtesting systems keep running a loop awaiting for the incoming market data ticks, candle bars, order transactions as events, and make buy or sell decisions based on these events. Quntopian is a successful commercial on-line backtesting service implemented by event-driven simulation. It listens to every incoming candle bars as events and exposes a python API for users to script their own event handler. Users can calculate technical indicators or place buy or sell orders or plot charts within this event handler. Other famous strategy backtesting platforms such as MetaTrader, PyAlgoTrade, QuantConnect also employ event-driven simulation. The main advantages of event-based simulation is that its logic is the same with manual live trading, therefore it eliminates the "Look-ahead Bias". And by connecting to real-time data stream and trading brokers, event-driven simulation can be directly used for live trading without any major code change. However, event-driven simulation significantly introduce much more complexity than vector-based simulation and slow down the execution.

2.2.2 Data storage design and Database selection

Strategy backtesting systems are supposed to handle and store huge amount of market data, mainly comprised of price and volume data of corresponding financial instruments. The basic format of market data is Trade-Quote data offered by exchanges or financial data vendors, which contains price and size of traded transactions and bid-ask information of trading orders. However, tick data is excessive in size and muddled for human understanding, therefore, for analysis purpose, tick data are always aggregated into candlestick bars. Candle-stick charts are the representations for the price movements of financial instruments in a time period. Figure 2.2, shows the details of candle-stick bar chart. In all, due to the great quantity of tick or candle-stick data, market data preprocessing and storage and is a vital factor for the execution efficiency of all the strategy backtesting systems.

The simplest method to store the market data is to save all the data in a CSV format file and load the data in one batch and do all the processing in memory. Disadvantages of this method are significant because of the temporally and spatially inefficiency. Therefore, appropriate database engines are needed to be selected and properly embedded into the design and implementation of

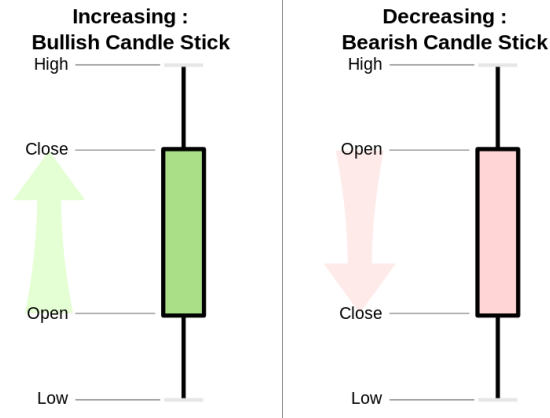


Figure 2.2: Definition of candlestick chart

strategy backtesting systems. Relational Database Management systems such as Oracle, MySQL, Microsoft SQL Server, PostgreSQL and SQLite are the most commonly choices for experimental or commercial strategy backtesting systems. Quantopian and PyAlgoTrade are both python based commercial strategy backtesting systems using SQLite as data storage. Both systems focus on low-frequency candle-stick bar based strategies, so there is no need for them to store and process large-scale tick data. SQLite also serves as database engine on TSS[14], a light-weight strategy backtesting system targeting at daily trend-following strategies, equipped with logistic regression classification. These backtesting systems employ relational database since relational databases support complex queries, better security, ACID properties, and are very convenient to be deployed on different environments. On the other hand, these systems only store candle-stick bar data, so the data volume is relatively small and the performance issue of relational databases is tolerable.

If a strategy backtesting system manages to store huge amount of tick data and aggregate the candle bars from tick data, NoSql databases such as MongoDB, Apache Cassandra, may be the best choices due to their high throughput, high scalability and distributed architectures. Rafique evaluated the performance of a NoSql database, Apache Cassandra compared with MySQL on writing and querying financial transaction data[46]. The study concludes that Apache Cassandra is strongly superior to MySQL since its much higher performance and its horizontal scalability over cluster nodes when processing historical financial data. Matt Kalan also produced a investigation which figures out that reasons why many banks or financial related firms use MongoDB as their

database engines for storing tick data is that MongoDB has characteristics of high throughput, flexible indexing, and MapReduce capability [9].

Among all the NoSql databases, time series databases are the database systems that is optimized for handling time series data, and the arrays of their data are indexed by time. The major advantages of time series databases are high throughput, high efficiency of time range queries and time aggregation queries, and excellent compression effectiveness [10]. Time series databases are commonly employed by many experimental and commercial strategy backtesting systems. AlgoTrader, which is a backtesting systems allow its user to use Java to develop complex intra-day algorithmic strategies, utilize InfluxDB as its main data storage. InfluxDB is a top ranked time series database which outperforms many other kinds of NoSql databases on writing and querying time series data[7][8]. Another famous strategy backtesting system, SmartQuant, even designs and implements its own time series database system, named QuantFile. QuantFile is an in-memory database, which only stores data in compressed binary format with time-stamp, and processes all the queries in memory.

2.2.3 Sequence Matching over Event Streams

One of the most important functions of strategy backtesting systems is to enable traders to model their trading knowledge as quantified strategies and detect trading signals of strategies. As mentioned in chapter 1, MTPS targets on backtesting strategies based on technical patterns and many the technical patterns are composed of technical indicator value changes with corresponding time orders. Therefore, MTPS transforms technical patterns into "technical event" sequences and employs algorithms to match these sequences over incoming event streams.

Algorithms of Event Sequence Matching over streams originates from regular expression matching and dynamic programming, which has been very well studied. One of the state-of-the-art algorithm for regular expression matching is FPGAs, which is a NFA based model that costs a $O(1)$ time and $O(n^2)$ space[47]. Inspired by FPGAs, the NFA^b model is proposed to handle the event sequence pattern matching[11]. The event sequence based patterns are similar with regular expressions since each event is just represented by a symbol character. NFA^b can model some logical relationships between events, such as Kleene closure and negation and their compositions. Augmented Finite Automaton(AFA) is a further extension of NFA^b , which can recognize

Feature	TradeStation	MetaTrader	Quantopian Zipline	PyAlgoTrade	CandleScanner
Programming API Support	✓	✓	✓	✓	×
GUI Support	✓	✓	×	×	✓
Portfolio Management	✓	✓	×	×	×
Build-in Indicators	✓	✓	✓	✓	✓
Pre-defined Strategies	✓	✓	×	×	✓

Table 2.1: Features of Strategy Designing Interface

dynamic patterns or handle the input stream disorder problem by augmenting the state of NFA with a register[21]. Recently, a more general model for event-based pattern detection called "Event Calculus"[15] was presented to deal with patterns composed by events both happening at some time instants and lasting in time intervals or patterns consist of events cascaded by more complicated predicates. However, considering the case of detecting technical analysis event-based patterns for market prediction, all the methods above are too abstract to apply on. Since the time intervals between technical events are critical for such patterns, the algorithm employed by MTPS for pattern detection should adopt the time intervals between events as a state transformation condition.

2.2.4 Strategy describing Interface

Strategy describing interfaces are either graphical user interfaces (GUI) or application programming interfaces (API) provided to users for transforming their trading knowledge into algorithmic trading strategies. A good strategy describing interface contains the major business logic of the backtesting system and will distinguish itself from others. In the present work, MTPS is a backtesting system focus on backtesting low-frequency technical analysis based trend-following strategies, therefore we summarize and compare the design of strategy describing interfaces of several most popular commercial technical analysis trend-following backtesting systems.

In Table 2.1, 5 core features of strategy describing interfaces of 5 famous commercial backtesting systems are investigated, the results present features included in the systems. Since strategy describing interfaces are for users to input their rules to make the buy or sell market decisions, they are either GUIs or programming APIs or mixture of both. Among the 5 systems being investigated, Quantopian Zipline, and PyAlgoTrade expose python APIs for users to handle candle bar arrival

events, while TradeStation provides a self-defined script language, EasyLanguage to calculate indicators or construct strategies on candle-stick charts. MetaTrader offers user its own programming language, MQL, which is the most powerful API allows users to interact with candle charts, technical indicators, portfolio status, and brokers actively. Programming API based describing interfaces can strongly enhance the flexibility of strategy logic, and the code of strategies might be easily to reused and extended. However, such kind of strategy describing interfaces require their users to have solid background knowledge on computer programming, and a comprehensive ability to translate the heuristic graphical trading idea into source code. On the other hand, GUI based describing interfaces are more user-friendly and do not require any coding. However, the drawbacks of existing GUI based interfaces are significant. For instance, the GUI of TradeStation, MetaTrader, CandleScanner only allow their users to configure some pre-defined strategies' parameters, while modifications of these strategies are not supported. Therefore, users can not customize their own strategies by GUI easily.

In addition to designing strategies, several important functionalities may also be included in powerful strategy describing interfaces. Portfolio management for strategy backtesting systems refers to backtest a branch of strategies concurrently and make market decisions not only based on trading signals, but the capital value and strategy performance. Portfolio management can help traders to monitor the overall status of investment and control trading risks. Amid the 5 systems above, only TradeStation and MetaTrader offer very simple portfolio management functionalities. They just allow users to run multiple strategies together and allocate an percentage of money on each strategies. However, such crude functionalities can not help users much on risk control. Last but not least, abundant build-in technical indicators and pre-defined strategies are also critical to assist users to design their own strategies. Users are able to combine and modify the build-in technical indicators and pre-defined strategies to form customized strategies. However, for all the existing backtesting systems, only very simple and well-known indicators and strategies, such as moving average crossover, RSI overbought, are provided, which doesn't improve the usability much.

CHAPTER 3

TECHNICAL ANALYSIS STRATEGIES

Technical analyses are heuristic mathematical analysis methods that take the historical market data (price and volume mainly) and give a prediction on whether the current trend of price movement will hold or reverse. Technical analyses root on the sentiments of traders about supply and demands in the market. These kinds of trading methodologies highly rely on the experiences and idea of traders and there are no much mathematical theories behind them. However, researches[19][50][23] have proven that even some mostly simple and well-known technical analysis based strategies can bring traders excessive profit on stock or foreign exchange markets. In this chapter, we will introduce some background knowledge about technical analysis including technical indicators, technical patterns and strategy evaluation to reveal designing principles of MTPS.

3.1 Technical Indicators

Technical indicators play the fundamental role of technical analysis. A technical indicator refers to a series of values calculated from historical market data such as price, volume, and open interest and used for predicting future prices. MTPS only includes the technical indicators derived from candlestick charts, for which each candle bar is associated with only one indicator value, or say each indicator value is assign to one time instant. MTPS categorize its indicators into two major classes, single-bar indicators and price-level indicators. Such categorization is based on how a indicator will be used in strategies. A single-bar indicator measures market sentiment by its absolute value, while a price-level indicator will trigger a market signal when the market price move close to or cross some specified price level. In the following subsections, the details of several most important indicators in MTPS will be briefly introduced.

3.1.1 Single-Bar Indicators

Definition 3.1.1 *Moving Average Convergence Divergence*

An *Exponential Moving Average(EMA)* is defined as,

$$EMA_t = \begin{cases} \alpha EMA_{t-1} + (1 - \alpha)C_t & : t > 1 \\ C_t & : t = 1 \end{cases}$$

where t denotes the t -th candle bar, E_t denotes EMA of t -th bar, C_t denotes the close price of t -th bar, and α is a constant smooth parameter.

Then the *Moving Average Convergence Divergence(MACD)* is defined as,

$$MACD = EMA_a - EMA_b$$

where $a < b$. Normally traders will choose $a = 12$ and $b = 26$.

MACD basically measures the differences between short-term momentum and long-term momentum for a market. If $MACD > 0$, it means the market price keeps increasing fast, and it triggers a bullish signal. While if $MACD < 0$ generates a bearish signal.

Definition 3.1.2 *Relative Strength Index*

Define \bar{U}_t be the moving average of upper close and \bar{D}_t be the moving average of lower close, that is,

$$U_t = \begin{cases} C_t & : C_t > C_{t-1} \\ 0 & : else \end{cases}$$

$$D_t = \begin{cases} C_t & : C_t < C_{t-1} \\ 0 & : else \end{cases}$$

therefore, the moving average are $\bar{U}_t = \sum_{i=1}^{i=t} U_i$ and $\bar{D}_t = \sum_{i=1}^{i=t} D_i$, and then define the *Relative Strength(RS)* and *Relative Strength Index(RSI)* is given as,

$$RS = \frac{\bar{U}_t}{D_t}, RSI = 100 - \frac{100}{1+RS}$$

In normal cases, the time range t is 14.

The scale of RSI is from 0 to 100, and a large value implies that the market has gone up a lot recently, while a small value implies that there market has gone down a lot recently. A RSI value greater than 70 indicates that the instrument has been over-bought, which means it is going to be bearish later. While a RSI value less than 30 indicates that the instrument has been over-sold, which means it is going to be bullish later.

Definition 3.1.3 *Stochastic Oscillator*

Define H_t be the highest price in last t candle bars, and L_t be the lowest price in last t candle bars, C be the close price of current bar. Then a stochastic oscillator $FastK$ is defined as,

$$FastK = 100(C - L_t)/(H_t - L_t)$$

the most commonly used value of t is 14.

then another stochastic oscillator $SlowK$ is the 3-bar moving average of $FastK$, and $SlowD$ is the 3-bar moving average of $SlowK$.

The interpretation of stochastic oscillators is similar with RSI, the main difference is that their over-bought range is above 80 and the over-sold range is below 20. Another powerful usage of stochastic oscillators, is to consider two stochastic oscillators together. For instance, if $SlowD$ cross above $SlowK$ at the over-bought range, then it becomes a very strong signal of bearish.

All the single-bar indicators can give traders clear ideas on whether the prices of financial instruments will go up or down in the future. However, such kind of indicators do not provide any quantitative information of market movements such as how much up or down trends will go and how long trends will last, which are not accurate enough for trading purpose. In the next subsection, price-level-indicators will be illustrated, which will indicate reference prices to serve as targets of entries or exits of strategies.

3.1.2 Price-Level Indicator

Price-level indicators generate prices or price ranges which are the psychological price targets that market price will move to in the future. Price-level indicators are even more valuable than single-bar indicators, however, it is more difficult to give clear definitions of price-Level indicators. Therefore, MTPS proposes and implements some heuristic price-level indicators which show their power empirically and will be defined in the following subsection.

Definition 3.1.4 *Simple Moving Average*

Define the simple moving average of last t candle bars, MA_t , as,

$$MA_t = \sum_1^t C_t, \text{ where } C_t \text{ is the close price of current candle bar.} \text{hung2013profitability}$$

Simple moving average is one of the most popular price-level indicators to measure the public psychological price of an instrument. Its power has been addressed by many traders and researches[31][27] and it is very easy to be used and customized. A candle bar closing above some important simple moving averages indicates a bullish market, while a candle bar closing below the some important simple moving averages indicates a bearish market. Multiple simple moving averages can be combined together, i.e. short-term SMA cross above long-term SMA is named as "Golden Cross" meaning a bullish signal, while short-term SMA cross below long-term SMA is named as "Dead Cross" meaning a bearish signal.

In addition to SMA, MTPS also includes two kinds of most commonly used price-level indicators that are Support and Resistance. The idea behind Support and Resistance is to obtain key price lines such that either price movement trends will reverse when the market price approaches the lines or the trends will go further when market price cross them. For instance, if the market moves from a higher price to a low price line, and then shakes near this line several times, and finally pulls up back to a significant higher price, then we say the market movements are supported by the price line. or say market "react" with a support level. Conversely, if the market moves from a lower price to a high price line, and then shakes near this line several times, and finally drop down again to a significant lower price, then we say the market movements are resisted by the price

line, or say "react" with a resistance level. Based on the properties of Support and Resistance, the "trading range breakout" strategies are developed and applied by many traders, and researches also discovered such strategies can achieve a good empirical performance[19][?]. However, there are still no precise mathematical definitions of Support and Resistance, and they are normally recognized by traders visually from candle charts. Therefore, in this thesis, we propose a novel definition of Support and Resistance and integrate it in MTPS strategy description interface so that users can easily take use of Support and Resistance in their strategies.

Definition 3.1.5 *Support Level*

Suppose price level p is a potential Support level, by given an active range threshold r , a candle bar at time t , is defined to "react" with support p if and only if the low of the bar, l_t , satisfy the condition that $l_t \leq p + r$, the open of the bar o_t , and the close of the bar c_t satisfy the condition that $\min(o_t, c_t) \geq p - r$. A candle bar at time t , is defined to "breakout" support p if and only if the close c_t satisfy the condition that $c_{t-1} \geq p - r$ and $c_t < p - r$. Figure 3.1 gives an example of a candle bar "react" with support.

For a look-back period start from time h to current time t and a bar at s , where $h < s < t$, denote its low price is l_s , and then from time s to time $t - 1$ there are at least k candle bars "react" the support on level l_s with an active range threshold r , also from s to $t - 1$, there is no any bar "breakout" the support on level l_s . Then a support level at candle on time t is identified, i.e. $\text{Support}_{t-h,r,k} = l_s$, where $t - h$ is the look back length, r is active range threshold, and k is the total number of react.

Definition 3.1.6 *Resistance Level*

Suppose price level p is a potential Resistance level, by given an active range threshold r , a candle bar at time t , is defined to "react" with resistance on p if and only if the high of the bar, h_t , satisfy the condition $h_t \geq p - r$, the open of the bar o_t , and the close of the bar c_t satisfy the condition $\max(o_t, c_t) \leq p + r$. A candle bar at time t , is defined to "breakout" resistance p if and only if the last close, c_{t-1} , and the current close c_t satisfy the condition $c_{t-1} \leq p + r$ and $c_t > p + r$.

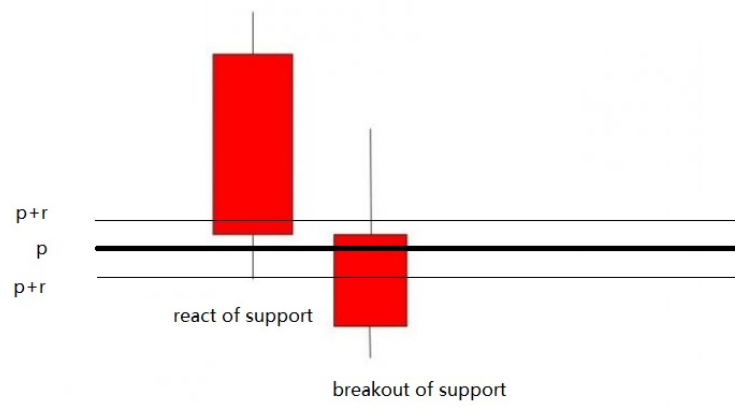


Figure 3.1: React and Breakout of Support



Figure 3.2: example of Support levels

For a look-back period start from time h to current time t and a bar at time s , where $h < s < t$, its high price is h_s , and then from time s to time $t - 1$ there are at least k candle bars "react" with resistance h_s with an active range threshold r , also from s to $t - 1$, there is no any bar "breakout" the resistance h_s . Then a Resistance Level at candle on time t is identified, i.e. $Resistance_{t-h,r,k} = h_s$, where $t-h$ is the look back length, r is active range threshold, and k is the total number of react.

The main idea of the above definitions are inspired by the graphical meaning of Support/Resistance. When a bar reacts with Support or Resistance, it means the price trend is blocked at this Support or Resistance level. The active range is a price interval around the Support and Resistance to avoid over-sensitive problems of using one single price line. The number k , denotes the number of reacts of the Support or Resistance, therefore a larger k means a stronger Support or Resistance, which means there is a stronger confidence of price trend reverse. Figure 3.2 shows an example of $Support_{20,1,2}$ on daily chart of gold price at Oct 2017.

3.2 Technical Patterns

In candle-stick charts, trading patterns refer to series of candle bars forming special graphical shapes. Similar with indicators, trading patterns show prediction power on the future price movements. Tremendous number of trading patterns such as Double Tops and Bottoms, Head and Shoulders, and Cup and Handle are employed by traders to forecast future instrument prices. However, many of these patterns can only be recognized graphically, and therefore are difficult to be quantified and not suitable for algorithmic trading. In this thesis, our main focus is a special kind of trading patterns called "Technical Patterns", which are series of price changes associated with changes of values of technical indicators. Compared with other kinds of trading patterns, "Technical Patterns" are more well-defined and easy to be interpreted by humans. In this section, we will describe two most famous technical patterns, and will show how MTPS allow users to describe their own technical patterns in later chapters.

Support turn resistance is a simple but powerful technical patterns that can be observed in many markets. The definition of support turn resistance is that after a strong support level is breakout at a candle bar, the price then pull back in next several bars, however the highest prices of these



Figure 3.3: example of Support Turn Resistance

several bars go very close to this support level but can not go higher, which means the support level becomes a resistance level. Therefore, if a strong support level is breakout, traders may have very good opportunities to place short orders at this support level. Figure 3.3 shows a typical example of support turn resistance on the Euro-USD market at fall 2015. It can be easily discovered that the support level 1.10957 which has been reacted twice at Sep 2015, was breakout at 22nd Oct 2015, and then turned to a resistance in the next several days.

Another most commonly used technical pattern is momentum indicator divergence. Momentum indicators are the technical indicators measure the price movements on either the up or down direction in short time intervals before. MACD, RSI and Stochastic Oscillators are all momentum indicators. The larger these indicator values are, the bullish the market is, vise versa. The momentum indicator divergence patterns are used to express market situations when the price and momentum indicators move towards different directions. For instance, a bullish RSI divergence pattern can be described as when the market price keeps going down while the RSI value starts to go up. If such a pattern being observed at the end of a long-time down trend, it forms a signal of bearish, which means the market is going to be reverse. Similar patterns such as bearish RSI divergence, bullish/bearish MACD divergence are often found before trends reverse.

3.3 Multi-Time-Frame Technical Analysis

The terminology "time-frame" in technical analyses means the time scope of the candle charts that technical indicators calculated from. Being Calculated from larger time-frames means technical indicators have longer term view of markets. For example, the 50 days moving average(Day-50) is at the daily time-frame and 50 hours moving average(hourly-50) is at hourly time-frame. Therefore, the Day-50 represents a longer term market sentiment than hourly-50. Apparently, time-frames are important consideration of technical patterns.

Multi-time-frame technical patterns is a category of technical patterns which include technical indicators from more than one time-frames. The main advantage of Multi-time-frame technical patterns is to consider the long-term time-frame market sentiments and short-term market sentiments together, which can make the price predictions more accurate and sensitive. For instance, if a small time-frame long signal is observed after a large time-frame long signal being observed, the large time-frame long signal then reinforces the small time-frame long signal and a long entry trading signal with very high confidence is generated. Constantly, if a small time-frame long signal is observed after a large time-frame short signal being observed, traders may think twice about the market sentiments and wait until the trend of price movements become more clear.

Large amount of multi-time-frame technical patterns have already been developed by professional traders. Fundora design a multi-time-frame technical pattern to trade stock of Holly Corp by taking use of indicators such as weekly support level, daily 20 moving average as well as daily Bollinger Bands[25]. Another example of multi-time-frame technical pattern is provided by Krivo, which try to short the EURUSD by taking use of the Stochastic Slow indicators from both daily time-frame and 4-hourly time-frame[32]. From the two examples, we can easily conclude that the multi-time-frame technical patterns is able to bring excess profit over single-time-frame technical patterns.

3.4 Strategy and performance evaluation

In previous sections of this chapter, examples of technical indicators and technical patterns can be described via MTPS interface are introduced. In the context of MTPS, one single technical indicator or combination of technical indicators or more complex technical patterns are denoted as generally as patterns. Then a strategy in MTPS can be defined as a set of trading rules that will go long or short with a user specified position size triggered by certain patterns. In MTPS, a strategy is composed of an entry rule, several exit rules, and several stop-loss rules. Each trading rule contains a signal pattern and a trading position size. The position size may be a fix value or a value determined by the amount of money allocated on the strategy. Entry rules will only be triggered when there is no any open position, while exit rules can only be triggered when there are open positions and will close all the open positions. A trade in MTPS is then defined as the process from zero position to open positions and then to zero position again. Stop-loss rules are special kind of rules that will close all the open positions immediately by market orders. Stop-loss, however, are not triggered by patterns, but triggered when the price move towards to the opposite direction in certain distance, which means the loss of trades have achieved at certain amount.

After running of backtesting, MTPS will output the transaction lists which show the details of all trades made by trading rules. Besides the transaction lists, MTPS employs several performance evaluation measurements for users to compare the return and risk of strategies. In this section, we will briefly introduce several measurements implemented.

1. Annualized Return Rate:

Annualized return rate is the geometric mean over the number of years of total return by initial capital. Annualized return rate is the most straightforward way to measure the strategy profitability. If r_a denotes the annualized return rate, y denotes the number of years, C denotes the initial capital, R denotes the total return(initial capital included), then the annualized return can be defined as,

$$(1 + r_a)^y \times C = R$$

2. Maximum Draw-down:

Draw-down refers to the strategy value(the value of holding position and remaining cash) difference between current minimum of strategy value and last local maximum of strategy value. Then the maximum draw-down is the largest draw-down value appears during the whole backtesting period. Maximum draw-down measures the largest risk being taken while running the strategies. Maximum draw-down is especially important while designing strategies for leverage trading. For instance, when doing futures trading, if the current draw-down exceeds a certain limit, the futures brokers may call margin from their traders, and if the traders can not pay for the extra margin, their open positions will be forced to close, which will lead a huge loss.

3. Sharp Ratio:

Sharp ratio is one of the most popular measure of strategy risk-adjusted return, which takes risk ,return and opportunity cost into consideration all together in one value. Suppose \bar{r} is the simple average over the return rate of each the day during backtesting, r_f is the risk-free investment return rate, σ is the standard deviation of daily return rate, the the formula of sharp ratio is,

$$SharpRatio = (\bar{r} - r_f)/\sigma$$

CHAPTER 4

MTPS STRATEGY DESCRIPTION SYSTEM AND GRAPHICAL USER INTERFACE

In Chapter 1, we mentioned that the major motivation of MTPS is to assisting traders to describe their own technical strategies via GUI and execute the strategies on data streams. In subsection 2.2.4, the strategy describing interfaces of several most popular backtesting systems are reviewed and summarized. It's easy to figure out that these describing interfaces are facing a fatal dilemma.

On one hand, if description interfaces of backtesting systems intend to allow users to describe advanced and complex logics in algorithmic strategies, they will require users to do some programming or scripting. For instance, TradeStation requires users to learn "Easy Language", MetaTrader requires knowledge about "MQL", and Quantopian needs users to learn Python programming. Such kinds of programming based strategy description interfaces are obstacles for traditional traders to use backtesting systems, especially for the traders who develop their technical trading knowledge before the wide spread of high-level programming languages.

On the other hand, there are also GUIs provided by backtesting systems for users to describe strategies conveniently. Successful commercial backtesting systems such as TradeStation, MetaTrader and NinjaTrader all promote users to describe their strategies via GUI based strategy description systems. However, in their description systems, users can only first select pre-defined strategies, and then configure the values of parameters of technical indicators used in the pre-defined strategies to generate their own strategies. Figure 4.1 shows the GUI window of NinjaTrader strategy describing interface. In this example, users are required to first choose a pre-defined strategy such as "Sample MA crossover", and can only change two parameters that is "Fast" and "Slow" to define their own strategies. However, if users would like to design technical strategies with logics totally different with the strategies in the pre-defined strategy list, they can not achieve this goal via the GUI. Therefore, MTPS is dedicated to solve the problem that allow users to describe and

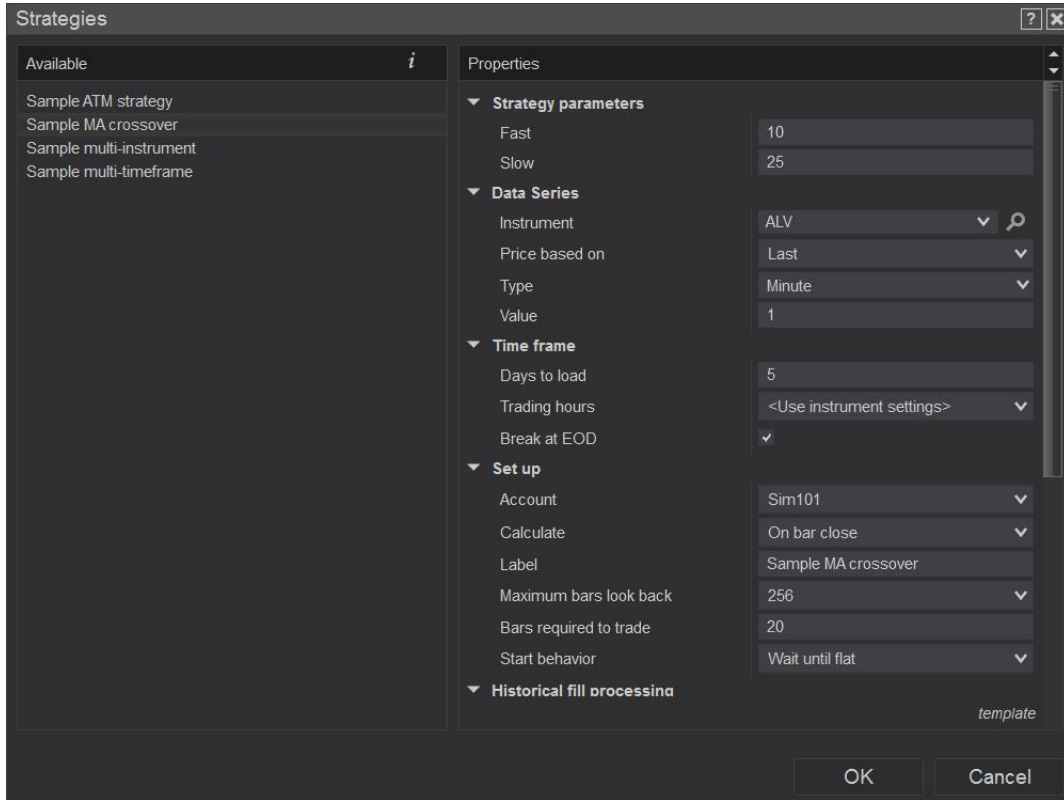


Figure 4.1: strategy description page of NinjaTrader

design algorithmic technical strategies via pure GUI. This purpose can be realized by introducing a 3-layer hierarchical MTPS strategy description system and their corresponding GUI.

The main principle of the MTPS strategy description system is inspired by an observation that many categories of technical patterns can be interpreted as sequences of characteristic candle bars within time windows. Such observation has already been addressed and justified by many researches. For instance, classical technical pattern "Head-and-Shoulders" is expressed as a sequence of special bars by Lo.[38]. Also, Leigh showed that another important technical pattern "Bull Flag" can be represented by bar sequence[34]. Based on this observation, the process of identification of technical patterns can be transformed into two steps, that are the identification of characteristic candle bars and the detection of sequences of characteristic candles in sliding time windows. Hereby, the MTPS strategy description system promotes users to describe their own technical strategies as sequences of characteristic candles via a 3-layer hierarchy as follows,

1. **Event Layer:** The terminology "Event" in MTPS refers to special charting or technical conditions of candle-stick bars. Users are allowed to create, modify and store their own events via the Event Layer GUI. Given a user defined event, MTPS Pattern Analyzer will select all the candle bars satisfy the conditions in the event. These candle bars filtered out by Event Layer are equivalent to characteristic candles mentioned above.
2. **Pattern Layer:** A pattern in MTPS consists of several events as well as time orders and time intervals between events. In the GUI of Pattern Layer, users can browse the events they created and saved in Event Layer and use them to compose new technical patterns. Users then, can select these events by their name and specify the time intervals between them via GUI. At the time of strategy simulation, the MTPS Pattern Analyzer will detect the appearance of the user defined patterns, and emit trading signals at candle bars on which the last events of patterns happen.
3. **Strategy Layer:** A strategy in MTPS is a composition of several trading rules, at least, an entry rule, an exit rule and a stop-loss rule. Each trading rule includes a signal pattern as well as an order action such as buy or sell, and the order position size. In the GUI of Strategy Layer, users can select patterns created and saved in Pattern Layer as the signal patterns of trading rules, and configure the order actions and order size. During the process of backtesting or real-time trading, MTPS will listen to all the signal patterns over the market data stream, and if there is any pattern triggers, the corresponding order will be sent to the order management system and trading brokers.

By employing the three layer hierarchy, MTPS promote users to create and describe their strategy layer by layer, and associates each layer with a GUI page. In addition, GUI page of simulation is provided in MTPS as a administration interface to trigger strategy backtesting. Figure 4.2 illustrates the overview of MTPS GUI flow, showing a screenshot of each GUI pages, and giving brief introductions to them. When designing technical strategies, users will first need to create necessary events on the Event-Page and save them with given names into the file system in local computer. Users can then load these events and edit them in the Event-Page also. After events being created, users will create patterns in the Pattern-Page via selecting the names of events in Event-Pages and

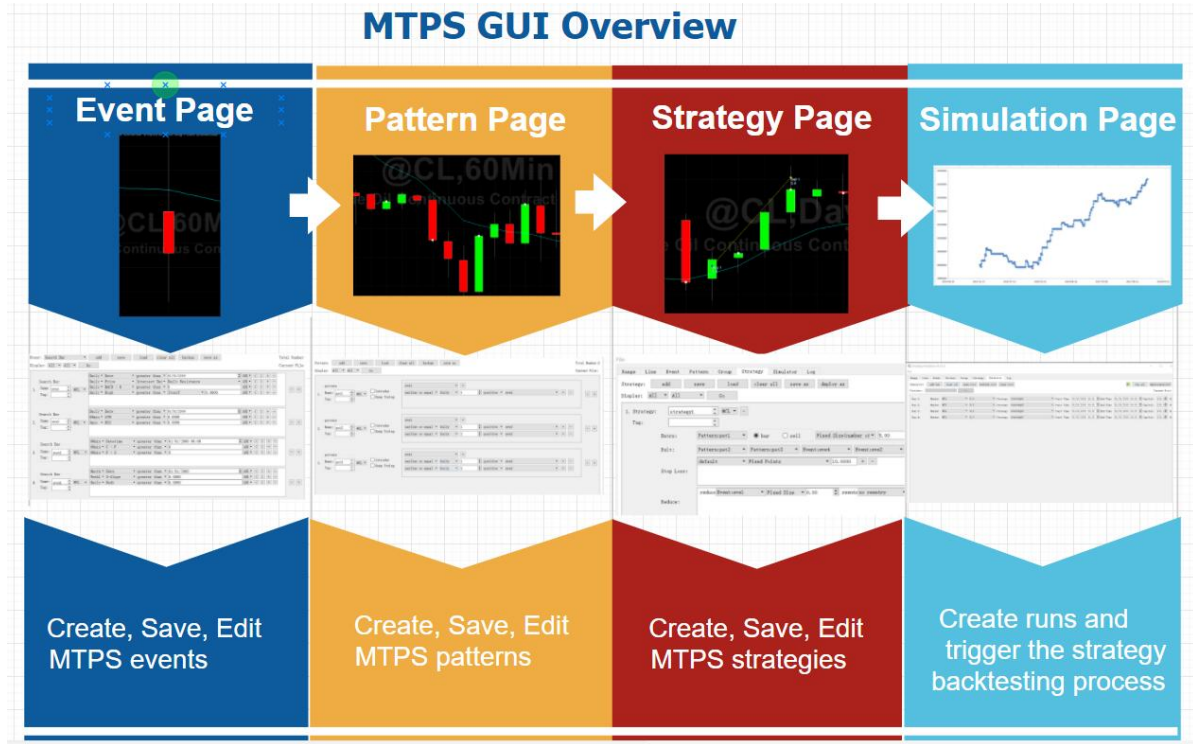


Figure 4.2: Overview of MTPS GUI flow

configuring the time interval between events. Patterns are also be able to be saved with names and loaded for later usage. The third step of user interaction of MTPS is to create strategies in Strategy-Page. Users can select the events or patterns presented in Event-Page and Pattern-Page as entry, exit or stop-loss signals in strategies. Users are also be able to save and load strategies in the file system of local computer. Finally, users can create runs on the Simulation-Page. Each run is a combination of a strategy with its backtesting start time and end time and a financial instrument. Users can select multiple strategies listed on Strategy-Page to create combination of runs on Simulator-Page. Until this step, users have completed the whole process of strategy description, and by clicking the "run all" button, users will trigger the backtesting process to simulate all their defined strategies on the selected time period and financial instruments.

There are three important advantages of MTPS strategy description system compared with the strategy description interfaces of existing strategy backtesting systems. First of all, the Event Layer and the Pattern Layer guide users to describe complex technical patterns with pure GUI and allow

users to use these patterns in algorithmic trading strategies. Event Layer ask users to specify conditions on selecting characteristic candle bars while Pattern Layer ask users to specify the time order and time intervals between characteristic candle bars. Therefore, as long as users can decompose their patterns into sequences of characteristic candle bars, they are able to describe these patterns in MTPS GUI. Comparing with the two categories of strategy description interfaces discussed at the beginning of this chapter, MTPS strategy description system provides much larger flexibility than the GUI allowing only slightly modification on pre-defined strategies, while requires much less learning efforts than the programming interfaces.

In addition, another important benefit of this hierarchical strategy description system is to make the building blocks of strategies reusable in other strategies. For instance, users may define an event containing information "Daily RSI greater than 70", save it, and then put it into several bearish patterns and develop different technical strategies. Such kind of detachable formats of technical patterns and strategies will reduce the effort of strategy developing.

Last but not least, users can easily integrate technical indicators of multiple time-frames in a single strategy to perform multi-time-frame technical analyses. In the backtesting systems introduced previously, only one time-frame of technical indicators can be assigned to one strategy. For example, NinjaTrader and MetaTrader treat time-frame as global variables of strategies. While, in TradeStation, each technical strategy is associated with a candle chart, and the time-frame of all technical indicators in that strategy is fixed by the time-frame of the candle chart. However, in MTPS, the time-frame of technical indicators are configured at Event Layer. Users can include indicators of different time-frames in one event or include events containing indicators of different time-frames in one pattern. Therefore, our goal to describe multi-time-frame technical strategies on GUI is achieved.

The following sections of this chapter will focus on elaborating the structure of each Layer of MTPS strategy description system, and showing how the GUI of each Layer can interact with users.

4.1 Event Layer

Definition 4.1.1 *Event* An event in MTPS is a one or several conditions of technical indicators or

charting to filter candle bars. In MTPS Pattern Analyzer, for a given newly coming candle bar as input, it will report events "trigger" as long as the bar satisfies the conditions of events.

Events are the most fundamental elements of MTPS strategies. They offer users capability to describe and find candle bars in special graphical shape or at the beginning or end of price movement trends. As discussed in chapter 3, MTPS mainly support two categories of technical indicators, single-bar indicators and price-level indicators. Therefore, most of the events can be described by MTPS are also basically divided into two kinds, "search-bar" events and "search-line" events.

Search-bar events are targeting at filtering out characteristic candles whose technical indicator values are located in some specific value ranges. A search-bar event in MTPS is described by one or several "technical filters". Each "technical filters" consists of a time-frame, an indicator name, a threshold value and a comparison predicate($\leq, \geq, <, >, =$), which will return a boolean value on each candle-bar. Then these "technical filters" are connected by logical predicates such as "And", "Or", which formulates a logical expression. This logical expression will be checked on each candle-bar during strategy execution. If the logical expression is true on the bar, this bar will be filtered, and such situation is called event trigger on the bar. The structure of a search-bar event is visualized on figure 4.3. In GUI of MPTS event-page, after the creation of search-bar events, users are able to configure every components of the events, that are time-frames, indicator names, comparators and logic predicates. Through this GUI, users are able to describe the characteristic candles in a human-understandable manner. Figure 4.4 shows a concrete example of a search-bar event on MTPS event-page.

The other important kind of technical event is search-line event which based on price-level indicators discussed in chapter 3. A search-line event, tries to find candle bars that have specific relationships with some pre-defined price-level indicators. To define a search-line event, users first need to define a price line set. The price line set is a set of price-level indicators, that are either Moving average, Support or Resistance with some user specified conditions. MTPS offers many conditions to filter price lines, for instance, "Distance" is the most commonly used conditions to filter Support and Resistance. Users may want to only retrieve support lines with condition $Distance > 1$. The attribute *Distance* for a support level denotes the price difference from this

time-frame 1	Indicator 1	comparator 1	threshold 1	logic predicate 1
time-frame 2	Indicator 2	comparator 2	threshold 2	logic predicate 2
time-frame ...	Indicator ...	comparator ...	threshold ...	logic predicate ...

Figure 4.3: search bar event structure

Figure 4.4: search bar event GUI

Figure 4.5: line-page GUI

level to the nearest lower support level. The larger Dis is, the stronger the support is. In this case, we only retrieve support level at price P_s with no any other support levels at price p , such that $p < P_s \leq p + 1$. In MTPS GUI, a line-page is employed for users to define conditions to select price lines that can be used for search-line event. The structure of the price line description in line-page is the same with the structure of search-bar event, and an example of line-page GUI is presented in figure 4.5.

The image shows a GUI for creating search-line events. It features a 'Seach Lines' section with a 'Name' field (containing 'sline1') and a 'Tag' field. To the right, there are several dropdown menus: '@CL', 'Daily', 'Price', 'Intersect', and 'line1'.

Figure 4.6: search-line event GUI

After price lines being defined properly, users are able to save these lines with names. Then users will take use of the saved price lines to create search-line events. A search-line event also consists of one or several "technical filters". Each "technical" filter is composed by a time-frame, a relationship, a price line name, and a threshold value. The relationship is used to filter candle-bars with special price relationship with price lines. For instance, the relationship "intersect" can be used to select candle bars with high price h_p , low price l_p and price line at price L , such that $l_p \leq L \leq h_p$. The threshold is used to specify tolerance of the relationship, for example, the relationship "intersect" discussed above with a threshold r describes the situation that $l_p - r \leq L \leq h_p + r$. Users are able to configure all the components of search-line events and select line names listed in line-page, the GUI of a search-line event is shown in figure 4.6.

Apart from search-bar event and search-line event, other types of events are also included in MTPS Event Layer. For example, "Displacement" event aims to filter candle-bars that move exceed an amount of displacement from last strategy entry price. All these types of MTPS events will be saved and used later in pattern-page or strategy-page. When running the backtesting, on each candle-bar sent into MTPS, the events will report True or False signals to inform the MTPS Pattern Analyzer whether the events trigger on the bar or not. These signals can be severed as market entry/exit signals or used to do pattern detection in the pattern layer.

4.2 Pattern Layer

As discussed at the beginning of this chapter, many technical patterns can be represented by sequences of characteristic bars. Therefore, MTPS Pattern Layer promote users to describe their technical patterns via time sequences of MTPS technical events. The definition of MTPS technical

pattern is as following,

Definition 4.2.1 Pattern *A technical pattern in MTPS is composed by a sequence of technical events associated with a time order and time intervals between events. A pattern is consist of a sequence of events $[E_1, E_2, \dots, E_n]$, and a sequence of time intervals T_1, T_2, \dots, T_{n-1} . Each time interval T_i is composed by a pair of time-frame and value. Pattern P triggers if and only if $\forall E_i \in [E_2, \dots, E_n]$, E_i triggers on or after the bar where E_{i-1} is found to trigger within time interval T_{i-1} .*

To illustrate the definition of technical pattern more clearly, we take a very simple three-event pattern as an example. Suppose there is a pattern P contains three events $[E_1, E_2, E_3]$ and two time intervals $[T_1, T_2]$, where T_1 is a pair "60min,3" and T_2 is also a pair "60min,6". When running strategies containing pattern P , MTPS Pattern Analyzer will try to find a sequence of candle-bars $[b_1, b_2, b_3]$ on 60min candle chart. Denoting $[t_1, t_2, t_3]$ is the close time index of $[b_1, b_2, b_3]$, then the sequence $[b_1, b_2, b_3]$ is a match of pattern P if and only if $t_2 - t_1 \leq T_1$ and $t_3 - t_2 \leq T_2$. If $[b_1, b_2, b_3]$ is a match of P , then we say P triggers on bar b_3 .

Since each single characteristic bar been selected by MTPS events indicates a long or short trend already, the reason why the MTPS pattern definition can represent real-life technical patterns well, is that if several technical events trigger in turn in within a relative short time period, then they are going to reinforce each others. Following the definition of MTPS pattern, MTPS provides user a pattern-page in the strategy describing GUI to create their patterns by choosing the event names listed in event-page and time intervals in sequences. The structure of MTPS pattern GUI is shown in figure 4.7, and in the remaining part of this section, we use the MTPS Pattern Layer to create the technical pattern "Momentum Indicator Divergence" mentioned in Chapter 3. We will select the most commonly used momentum indicator "RSI", design a pattern on 60min chart, which is the "Hourly RSI Divergence" pattern.

The "Bullish Hourly RSI Divergence" tries to find double-bottom shapes on the 60 minute chart during 24 hours. As discussed above, the first step of describing technical patterns is to find out the characteristic bars. In this pattern, the four characteristic bars are the first bottom bar, the first

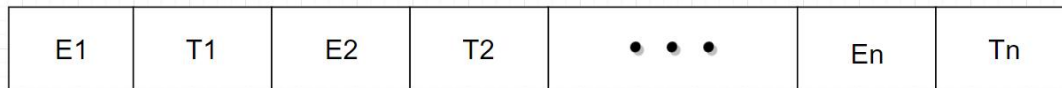


Figure 4.7: pattern layer structure

recovery bar, the second bottom bar and the second recovery bar. Therefore, we will first need three MTPS events to describe the three characteristic bars by MTPS events, that are,

1. The first search-bar event, event Bottom1, contains one technical filter, that is " $hourlyRSI < 30$ ". This event indicates that the price of the instrument dropped a lot recently and there is coming to be the first bottom of the market.
2. The second search-bar event, event Recovery1, also contains one technical filter, that is " $hourlyRSI \geq 30 + r$ ". There is a parameter " r " used for describing how much the RSI value recover from the first bottom.
3. The third event, event Bottom2, is a little more complicated, it contains three technical filters, that are "hourly, RSI, greater than, lowest RSI in last 24 hour", "hourly, low price of bar, less than, the lowest price of last 24 hour", "hourly, $RSI \geq 30$ ". This event describes a situation of the second bottom of the trend, and this bottom has a lower price than the first bottom, while its RSI is greater than the first bottom, which is a divergence.
4. The fourth event, event Recovery2 can be exactly the same with event Recovery1, to indicate the second bottom terminate.

The second step is to consider the pattern "Hourly RSI Divergence" as a sequence of the above four events, that is "Event Bottom1 triggers, and then within 12 hours Event Recovery triggers, and then within 12 hours Event Bottom2, and then within 6 hours Event Recovery2". Figure 4.8 shows how exactly to define this pattern in pattern-page GUI. By saving this pattern with a short name "60rsidiv", it can be used as an entry pattern in long strategies or an exit pattern in short strategies.

Indicator	Operator	Time-frame	Value	Condition	Recovery Event
Bottom1	earlier or equal	60min	12	positive	Recovery1
Bottom1	earlier or equal	60min	12	positive	Bottom2
Bottom1	earlier or equal	60min	6	positive	Recovery2

Figure 4.8: pattern 60rsidiv

4.3 Modeling Multi-time-frame Pattern

As mentioned in the contribution of the thesis, the capability of describing multi-time-frame technical patterns is one of the most distinguishable features of MTPS. After elaborating the details of event layer and pattern layer, we are able to summarize the two methods of MTPS for users to describe multi-time-frame technical patterns.

The first method is to employ multi-time-frame search-bar events. As mentioned in previous section, a search-bar event may contains multiple "technical filters", and each "technical filter" consists of a time-frame. Therefore, if users configure the "technical filters" in one event to have different time-frames, then multi-time-frame search-bar events are created. For instance, if a search-bar event contains two "technical filters" that are "Daily, RSI, greater than, 70" and "60min, MACD, less than, 0", then this search-bar event will look for candle-bars in 60 minute chart, whose MACD is less than 0, and then select bars also belonging to daily bars whose RSI greater than 70. In this search-bar event, users try to use bearish candle-bar from daily and 60min time-frame together to reinforce each other.

The second method is to use events of different time-frames to compose one pattern. Since each event contains at least one technical indicator, therefore, if at least two of the events used in one pattern contains indicators on chart of different time-frames, the pattern is then a multi-time-frame pattern. For example, users are able to describe a pattern in MTPS Patter Layer that is after a daily bar with $RSI \geq 70$ is found, an hourly bar $MACD < 0$ is observed. This pattern includes both daily indicator and hourly indicator, and hereby is a multi-time-frame pattern.

In conclusion, the structure of MTPS event layer and pattern layer empower users to involve

technical indicators on candle-bars of different charts, which is the capability of modeling multi-time-frame technical patterns.

4.4 Strategy Layer and Simulation-Page

After the creation of events and patterns, it is very natural for users to take use of them to describe their technical strategies. In MTPS, strategies are modeled as a set of "trading rules" defining the behaviors of strategies. Each trading rules describes a situation that if some signals are observed, orders will be sent to brokerage to entry or exit trades. Such trading rules simulate the behavior of human technical traders, and in MTPS, technical strategies contain at least three kinds of trading rules, that is "Entry Rule", "Exit Rule", and "Stop Loss Rule" and their functionalities are listed as follows,

1. **Entry Rule:** Entry rules empower strategies go from flat position to open positions(long/short). In MTPS, if a strategy holds no position, then only its entry rule is active. That is, on a candle bar, if strategies hold no position, only the patterns of the entry rule will be checked, and if the patterns trigger, buy/sell orders placement request will be sent to the order management system. Entry rules try to foreseen the long or short trends around the corner, and go long or short to follow them.
2. **Exit Rule:** Exit rules empower strategies go from open positions to flat position. In MTPS, only when strategies holds open positions, the Exit rules will be active. Exit rules in most of the cases, define the profit target of a corresponding entry, and go flat to realize the profit.
3. **Stop Loss Rule:** Stop loss rules will enforce strategies go to flat position when the loss of strategies exceed a certain amount. Stop loss are also only active when strategies hold open positions. However, unlike Exit rules, Stop loss rules are not driven by signal patterns, but driven by the amount of loss of strategies. Users are able to configure the loss threshold as fix amount which is an tight stop loss. Or users can specify to monitor a loss after the market price go up/down break a certain price-level indicator, which is a kind of trailing stop in trading theory. Stop Loss rules are the most important components of risk control of MTPS.

Entry Pattern	Buy/Sell	Position Size
Exit Pattern 1	Exit Pattern 2
Stop Loss Pattern 1	Stop Loss Pattern 2

Figure 4.9: strategy-page layout

Strategy: @CL -

Tag:

Entry: Pattern:pat1 ☒ buy ☐ sell Fixed Size(number) 1.00

Exit: Event:eve4 none none none none

Stop Loss: default Fixed Points 1.0000 + -

Figure 4.10: MTPS strategy-page GUI

All types of trading rules contains a "signal pattern", and for an "Entry Rule", there must be a "position size" associated with it to specify how much amount of the financial instrument to be invested on each entry. MTPS will send orders to brokerage to entry or exit a trade as long as the "patterns" trigger. In MTPS users are able to create strategies on strategy-page GUI by configuring the "signal patterns" of each trading rules with pattern names listed in pattern-age and specifying a "position size" for the entry. Figure 4.9 shows the layout of the MTPS strategy-page GUI, and figure 4.10 gives of an concrete example of the page.

Last but not least, MTPS strategy description GUI provides users a simulation-page to run the

Run 1:	Market	@CL	0.2	Strategy	strategy1	Start Time	01/01/2015 00:00	End Time	01/01/2017 00:00
Run 2:	Market	@CC	1	Strategy	strategy2	Start Time	01/01/2016 00:00	End Time	01/01/2018 00:00

Figure 4.11: Simulation Page GUI

backtesting process of one strategy or a combination of several strategies listed in strategy-page. In the simulation-page, each strategy will be associated with a financial instrument name, a start-time and an end-time. By clicking the "run" button, every strategies will be executed on the specific financial instruments over the specific time intervals. Figure 4.11 gives an example of running two strategies on different instruments over different time. After execution of all the strategies being completed, MTPS will generate "performance report", "order transaction list" and "equity curve" for users to evaluate their strategies.

4.5 Expressive Power and Limitations of MTPS Pattern Description

In the previous sections of this chapter, we illustrated the details of how MTPS users to describe their technical patterns in a "event-pattern-strategy" three layer hierarchical description system. We address that this MTPS strategy description system provide users much stronger flexibility of expressing technical patterns and strategies than all the other GUI solutions. We also gave several examples of using MTPS to describe typical technical patterns such as "RSI Divergence". However, the scope of patterns can be modeled by MTPS is still not clear. Therefore, in this chapter, we will explore the theoretical expressive power of MTPS pattern description and investigate several limitations of MTPS.

As presented in section 4.1 and section 4.2, technical patterns will be recognized by time sequences of technical events. Such process of pattern recognition can be also regarded as non-deterministic finite automaton (NFA), in which each state can represent a pattern candidate and the state transition rules are the events triggering or pattern candidates expiring. For example, consider a pattern containing 3 events $[E_1, E_2, E_3]$, a graph of NFA of this patterns is shown in figure 4.12. In this NFA, a starting state S_0 denotes no any pattern candidates found, while a intermediate S_i

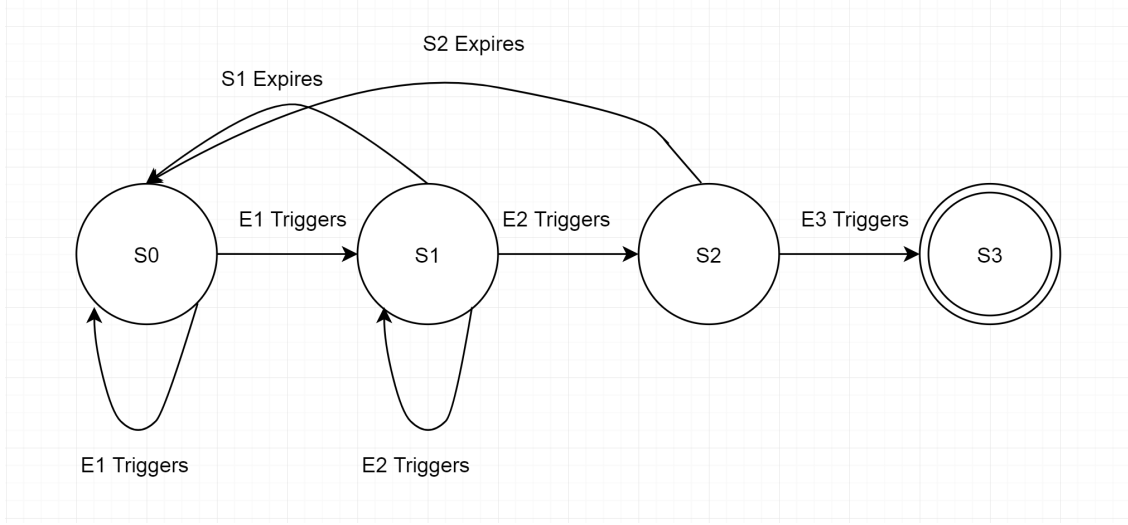


Figure 4.12: An example of MTPS pattern as NFA

denotes a pattern candidate containing events $[E_1, \dots, E_i]$, and S_3 is the accept state indicating the pattern triggers. If the NFA is at any state S_{i-1} , and on event E_i triggering, the automaton will transit from S_{i-1} into state S_i . In addition, at any state S_i , if the corresponding pattern candidate expires, the NFA will transit into the starting state S_0 .

From the analysis above, we can expand the this NFA model to recognize all the technical patterns that are able to be described in MTPS by assigning states to all the possible permutations of all technical events. In other words, if we denote all the events being able to be described in MTPS as an alphabet Σ , and denote events triggering and pattern candidates expiring are the production rules, then MTPS pattern description can be formulated as regular expressions. Therefore, expressive power of the patterns have the concatenation property of regular expression. Denote \mathbb{P} , then $\forall P_1, P_2 \in \mathbb{P}$, a pattern trying to look for situations that is after P_1 triggers, within a time period P_2 triggers is able to be described in MTPS. In general, we can conclude that the expressive power of MTPS pattern description is the same with regular expression or say NFA.

Additional to the expressive power, we will also present several limitations of MTPS pattern description. The current implementation of MTPS only includes limited number of technical indicators, so not all of the characteristic candles are able to be modeled in MTPS. However, even assuming that sufficient technical indicators are included in MTPS, there are still several limitations

of MTPS pattern in terms of expressive power as follows.

Firstly, if the chronological order of the characteristic candles of the patterns are not clear enough or the time period limits between the characteristics candles are not clear, users are not able to describe the patterns in MTPS. For instance, consider a pattern "sell short if there are 3 green daily bars in next 10 trading days". The characteristic candles are "green daily bar", however, there is no exact time period limits between these three characteristic candles, so users are not able to describe it in MTPS.

Secondly, technical patterns in MTPS only consider the time relationship between characteristic candles, however, many technical patterns will also take price relationship or indicator relationship between bars into consideration. For instance, a pattern "sell when the 10 day moving average of today is higher than a daily support of a previous trading day with $RSI > 70$ ", since this pattern requires comparison between SMA and Daily Support which are two price levels, users can not describe this pattern in MTPS.

Thirdly, it is all known that regular expressions are subsets of context free grammar, nonetheless, some technical patterns should only be active within some context. Such kinds of patterns are not able to be backtested in MTPS. For example, users may want to backtest a strategy that is "Sell short if price greater 50, while after the gross loss is greater than 10%, only sell short if price greater 55". The strategy contains two entry conditions, and the second condition is only active within context that "gross loss is greater than 10%", which can not be simulated in MTPS.

In conclusion, MTPS provide users a GUI based pattern description and detection solution. In MTPS strategy description system, users will gain much stronger flexibility than all the state-of-the-art strategy backtesting systems. The processes of describing MTPS patterns are similar with running NFA or producing regular expression strings. Therefore, the expressive power of MTPS is the same with NFA or regular expressions. On the other hand, there are still at least three types of technical patterns can not be described in MTPS which are limitations of MTPS.

CHAPTER 5

MTPS DESIGN AND IMPLEMENTATION

The goal of MTPS is to build a technical patterns based strategy backtesting system that allow users to design and backtest their customized strategies via a pure GUI, and deploy them to real-time trading. To achieve such a goal, MTPS is required to handle three major inputs, that are user designing strategies from GUI input, market data feed from data vendors, and brokers' response of order execution status. On the other hand, MTPS shall provide users at least two outputs, which are the market data candle-stick charts for users to make decisions, trading transactions and strategy performance for users to evaluate their strategies. Besides the functional requirements, MTPS also need to fulfill performance requirements, which is to detect technical patterns in a short time after new market data arrival. In figure 5.1, the details of architecture of MTPS is illustrated. The major components of MTPS and roles they played are stated as following,

1. **Data Updater:** The Data Updater will periodically scan data files downloaded from data vendors. When there is any market data update, it will read in the data, construct new candle bars or update current candle bars, and then push the updated bars into a message queue. Each of the new generated or updated bars will serve as data events and drive one iteration of strategies execution.
2. **Pattern Analyzer:** The Pattern Analyzer can translate users' GUI input into algorithmic patterns and strategies, and keep listening data events. When there are new data events, Pattern Analyzer will efficiently check whether there is any user defined patterns triggered, if so, it will generate trading signals to the Trading Engine.
3. **Trading Engine:** Trading Engine is the central controller of the whole process of backtesting or real-time trading. It will forward necessary messages from one components to the other and collect the feedback from them. It closely cooperates with Pattern Analyzer to make market decisions during strategy running.

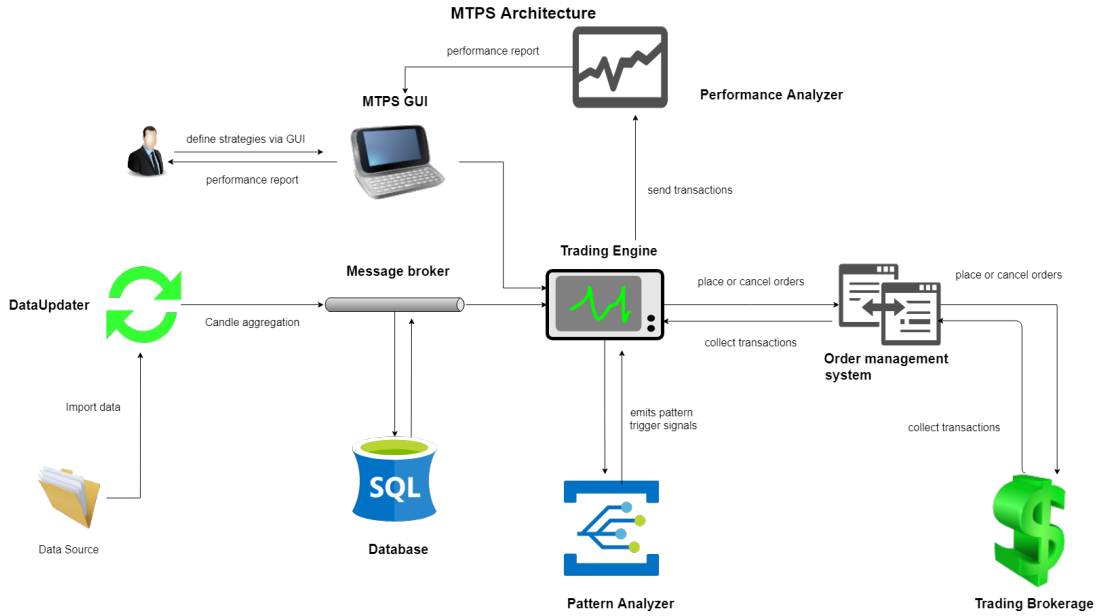


Figure 5.1: MTPS architecture

4. **Order Management System:** Order Management System will get pattern triggering signals from trading engine, and generate orders to trading brokers. It also retrieves feedback from brokers, such as transactions and margin requirements, and then returns these feedback to trading engine to update the status of strategies and analyzes strategy performance.
5. **Performance Analyzer:** Performance Analyzer will collect the order execution transactions from Trading Engine and generate performance analysis report to users. When doing the real-time trading, it will also monitor the total capital and all holding positions and print them to users.

In the following sections of this chapter, we will present details of how MTPS being designed, including the functionalities of different components and how they are organized together. To verify the power of MTPS design, we conduct an prototyping implementation of MTPS. In this implementation, C++ and Qt libraries are employed to build the architecture and core functions. In addition, we choose TradeStation as market data provider and simulation results visualization tool, SQLiteDB as database storage, and Interactive Broker as trading broker.

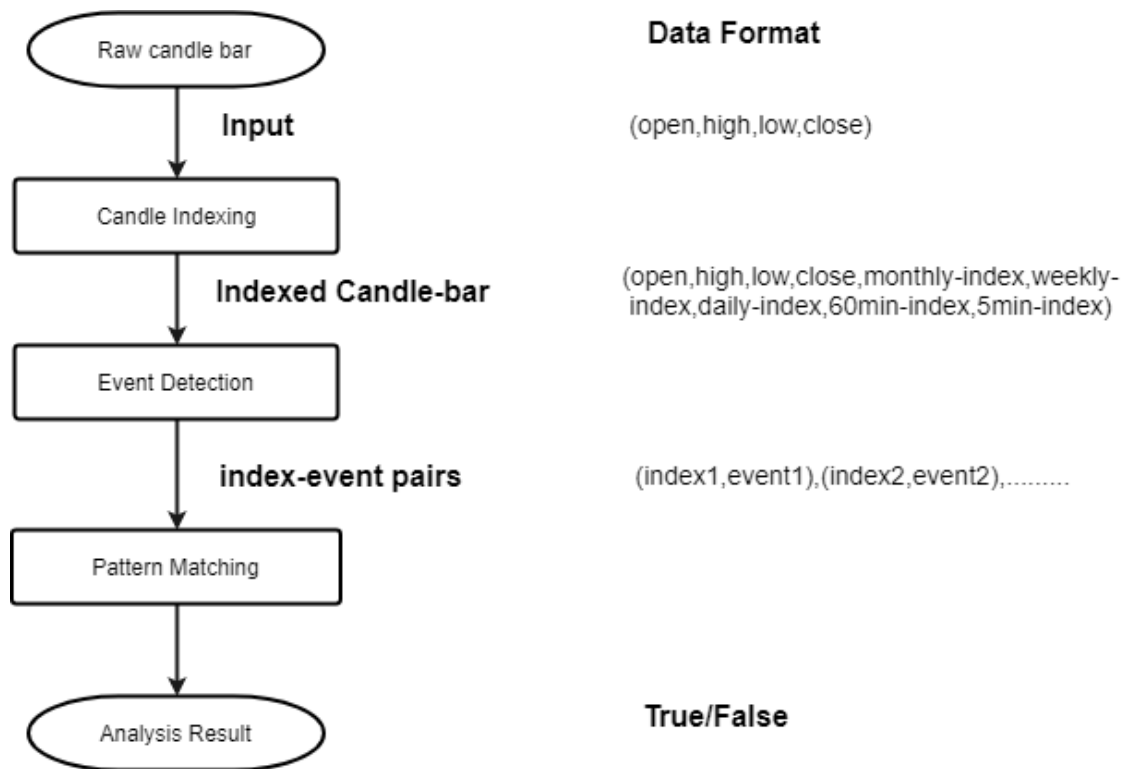


Figure 5.2: MTPS work-flow

5.1 MTPS Pattern Analyzer

Pattern Analyzer is the core component that contains the major novelty of MTPS and distinguish it from all the other commercial strategy backtesting systems. Combined with the event-layer and pattern-layer of MTPS strategy description model, Pattern Analyzer is able to match the MTPS technical pattern over a candle-bar stream, and then emits signals to drive the whole backtesting process. The whole process of MTPS pattern analysis can be decomposed into three steps that are "Candle Indexing", "Event Detecting" and "Pattern Matching". The design and implementation details of these three steps are presented in figure 5.2 and illustrated in the following subsections .

5.1.1 Multi-time-frame Candle Indexing

Candle indexing refers to the process of giving every candle-bars time indexes to represent their chronology order in a given time-frame. In each time-frame, the time index is just an integer start from 0 and increased by 1 on each candle-bar. For instance, if our database only contains one year of candle bars(252 daily bars), then the daily index of these daily candle bars will be from 0 to 251. During the running of system, if a new daily bar is sent into Pattern Analyzer, this bar will be given a daily index as 252. In MTPS, the time indexes will be regarded as a basic property of candle-bar and be saved into database persistently. With the indexes of bars, the time difference between two candle bars can be easily calculated by subtracting index of one bar from the other.

Another most important function of candle index is to keep a mapping between bars at the same close time on different time-frames. Therefore, each candle-bar will be given not only a time index of its own time-frame, but time indexes of all time-frames grater than its own. In our prototyping implementation of MTPS, we introduce 5 time-frames in our system, that is "monthly", "weekly", "daily", "60min", "5min". For example, a daily candle in the Candle Indexing step will be indexed by daily-time-frame, weekly-time-frame, and monthly-time-frame, corresponding to the trading day, trading week, and trading month the bar belongs to. In this example, if a daily candle is indexed as (*daily*, 252), (*weekly*, 52), (*Montly*, 12), it means that this daily bar is the 252th daily bar in the system, and it belongs to the 52th weekly bar and the 12th monthly bar. In summary, after Candle Indexing, a raw candle bar will be transformed into an indexed candle-bar that be used as event detection and pattern matching.

5.1.2 Event Detection

After a new candle-bar being indexed, all the technical indicators included in the events used in the running strategies will be calculated and these events will then be checked to trigger on this candle-bar or not, which is called the step "event detection". Before describing this process, we need to introduce the "event-time-frame" concept in MTPS. As mentioned before, a search-bar event may contain several "technical filters" and each "technical filter" consists of a time-frame. Therefore a search-bar event may contain multiple time-frames, and we define the minimum time-frame of the

event as the "event-time-frame". On the other hand, a search-line event only consists of a single time-frame, so this time-frame will be the event-time-frame. As long as an indexed candle-bar generated or updated by DataUpdater, the MTPS Pattern Analyzer will start detection process on these events.

Consider that a user may define a technical pattern P with events $[E_1, E_2, \dots, E_n]$ and time intervals $[T_1, T_2, \dots, T_{n-1}]$ in MTPS pattern-page GUI, and would like it to be recognized by MTPS Pattern Analyzer, then in this step, each event in $[E_1, E_2, \dots, E_n]$ will be checked to trigger on the current bar or not. From Chapter 4, we know that each event in MTPS contains one or several "technical filters", and each technical filters is related with a technical indicator. Therefore, to detect technical events, all the values of their related technical indicators on the current candle bar must be first calculated. With these calculated indicator values, every "technical filters" will be checked to be true or false, and then every event can be checked as triggering or not. Eventually, the output of the "event detection" step is a list of "index-event" pairs, which are created from all the events trigger on the current bar. The indexes in these "index-event" pairs are the bar indexes of the current candle-bar on the corresponding time-frames.

5.1.3 Pattern Matching

Since MTPS model technical patterns as sequences of characteristic candles, and the events in MTPS are representation of characteristic candles, the third step of MTPS pattern analysis is to match the user-specified sequences of events from event streams which is called "Pattern Matching". This step will take the streams of "index-event" pairs as input and generate signals about whether patterns triggering on the current bar or not.

We propose a high efficient "Fast Pattern Matching" algorithm to recognize sequences of events from streams of "index-event" pairs. Before introducing our "Fast Pattern Matching" algorithm, we will first elaborate the brute-force recursive sequence matching algorithm. Given the notation that a target pattern P with n events $[E_1, E_2, \dots, E_n]$ and time interval T_1, \dots, T_{n-1} , we want to find that in a stream of "index-event" pairs whether there are n "index-event" pairs (Idx_1, E_1) , (Idx_2, E_2) until (Idx_n, E_n) , such that $Idx_2 - Idx_1 \leq T_1$ until $Idx_n - Idx_{n-1} \leq T_{n-1}$. Denote pattern candidate P_i of pattern P , where $0 \leq i \leq n$, is a prefix of pattern P which contains i

events $[E_1, E_2, \dots, E_i]$ and intervals T_1, \dots, T_{i-1} satisfy conditions that $Idx_2 - Idx_1 \leq T_1$ until $Idx_i - Idx_{i-1} \leq T_{i-1}$. The brute-force pattern matching algorithm by finding pattern candidates backward recursively will be described below.

In order to perform pattern matching, all the input "index-event" pairs are recorded and restructured into a look-back table which contains bar indexes and their corresponding events. Denote I as the index of candle bar and \mathbb{E}_I as the set of all events in the look-back table at index I . Also for a pattern candidate P_i , denote L_{i+1} be the index of bar that event E_{i+1} triggers on. Algorithm 1 illustrates the details of FINDPA.

Algorithm 1 Recursive Pattern Matching Algorithm

```

1: procedure FINDPA( $P_i, I$ )
2:   if  $i$  is equal to  $n \wedge E_n \in \mathbb{E}_I$  then
3:     return FINDPARECURSIVE( $P_{n-1}, I - 1, I$ )
4:   return False
5: procedure FINDPARECURSIVE( $P_i, I, L_{i+1}$ )
6:   for  $Idx$  from  $I$  to  $0$  do
7:     if  $L_{i+1} - I \leq T_{i-1}$  then
8:       if  $E_i \in \mathbb{E}_I$  then
9:         if  $i$  is equal to  $1$  then
10:          return True
11:        else
12:          return FINDPARECURSIVE( $P_{i-1}, I - 1, I$ )  $\wedge$  FINDPARECURSIVE( $P_i, I -$ 
13:             $1, I$ )
14:        else
15:          return False

```

There are two procedures presented in Algorithm 1. Procedure "FINDPA" will be called on each input "index-event" pairs, and if the last event of P that is E_n is detected on a bar with index I , then the recursive procedure "FINDPARECURSIVE" will be called to start to find pattern candidate P_{n-1} by scanning back in the look-back table from index $I - 1$. During finding pattern candidate P_i , on the detection of event E_i , the "FINDPARECURSIVE" will go to two branches. One branch is to ignore the event and keep on scanning back to find P_i , while the other branch is to recursively call "FINDPARECURSIVE" to find P_{i-1} on the look-back table from index $I - 1$. Therefore, on each time of event E_i is detected, the size of the pattern matching problem is reduced by 1 and the recursive function will be called twice. In the worst cases, event E_i can be detected on all the bars and it is easy to give a simple worst case analysis of time complexity of this recursion.

Consider N as the maximum bar length of pattern P , that is $N = \sum_{i=1}^{n-1} T_i$, the time complexity of procedure "FINDPARECURSIVE" is equivalent to a recursion R , that is $R(N, n) = R(N - 1, n - 1) + R(N - 1, n)$. Since in most practical cases $n \ll N$, so the time complexity of recursive function is $O(2^N)$, which is exponential to the maximum bar length of the pattern.

In real-time trading, a high speed of signal pattern matching will significantly reduce trading cost, and therefore, the performance of the brute-force recursive procedure is insufficient for real-time intra-day trading. Hereby, we propose a "Fast Pattern Matching" algorithm in MTPS to address this performance issue. Inspired by the idea of Smith-Waterman algorithm[48] and some *NFA* based pattern matching algorithms such as FPGAs[47] and *NFA*^b[11], our algorithm also records the all the intermediate results of pattern matching. In dynamic programming based string matching, the algorithm stores all the intermediate results in a matrix. While our "Fast Pattern Matching" algorithm will keep intermediate results in a "pattern-candidate-map", which contains each events in pattern P as keys and the list of bar indexes that events trigger on. Denote the "pattern-candidate-map" as PCM and the bar indexes list of key E_i as $PCM[E_i]$. For the current candle bar with index I , consider a pattern candidate P_i with L as index of its last event E_i , we define P_i is feasible if $I - L \leq T_i$, which means this pattern candidate is still possible to grow into a completed pattern. The main idea of "Fast Pattern Matching" algorithm is to store all the feasible pattern candidates in PCM , and on each candle bar arrival, the algorithm will decide whether to grow or remove pattern candidates in PCM .

Algorithm 2 shows our detailed implementation of "Fast Pattern Matching" idea. Notice that in MTPS pattern matching, our target is only to find out whether a pattern P triggers on the current candle bar, instead of figuring out exactly candle bars that form the pattern P . Therefore, in PCM , we don't need to store whole pattern candidates. Instead, for a feasible pattern candidate P_i , only the index of its last event E_i is stored in PCM , in the list $PCM[E_i]$. By employing this trick, the pattern matching process can be split into two procedures, that is "FindPaFast" and "CheckFeasibility". For a input index-event pair (I, E_i) , the "FindPaFast" procedure will only check if there is any feasible pattern candidates P_{i-1} , that is whether PCM contains the index list with key E_{i-1} . If so, a new feasible pattern candidate P_i is detected and index I will be stored into $PCM[E_i]$. In the special case, if an index-event pair (I, E_n) being input, and if there is a feasible pattern candidate

Algorithm 2 Fast Pattern Matching Algorithm

```
1: procedure FINDPAFAST( $I, E_i$ )
2:   if  $PCM$  contains key  $E_{i-1} \wedge i > 1$  then
3:     if  $i$  is equals to  $n$  then
4:       return True
5:     else if
6:       if  $PCM$  contains key  $E_i$ 
7:         Add value  $I$  into list  $PCM[E_i]$ 
8:       else
9:         Add key-value pair ( $E_i$ , empty list) into  $PCM$ 
10:        Add value  $I$  into list  $PCM[E_i]$ 
11:   else if  $i$  is equals to 1 then
12:     if  $PCM$  contains key  $E_1$  then
13:       Add value  $I$  into list  $PCM[E_1]$ 
14:     else
15:       Add key-value pair ( $E_1$ , empty list) into  $PCM$ 
16:       Add value  $I$  into list  $PCM[E_1]$ 
17:   return False
18: procedure CHECKFEASIBILITY( $I$ )
19:   for each  $E_k$  in keys of  $PCM$  do
20:     if  $PCM$  contains key  $E_k$  then
21:       for each bar Index  $L \in PCM[E_k]$  do
22:         if  $I - L \geq T_k$  then
23:           Remove value  $L$  from  $PCM[E_k]$ 
24:       if  $PCM[E_k]$  is empty then
25:         Remove key-value pair ( $E_k, PCM[E_k]$ ) from  $PCM$ 
```

P_{n-1} such that $PCM[E_{n-1}]$ is not empty, it means that a completed pattern P is matched and a pattern triggering signal will be emitted. The second procedure "CheckFeasibility" will be called on each candle bar and take the bar index as input. It will scan all pattern candidates and check the feasibility of them, and for any pattern candidate P_k with index L of last event E_k , if $I - L > T_k$, then P_k is not feasible any more and will be removed from $PCM[E_k]$. If in list $PCM[E_k]$, all the elements have already been removed, the key E_k will be also removed from PCM .

We will give a simple complexity analysis of "Fast Pattern Matching" algorithm here. Remember the notation above is that N is the maximum bar length of pattern P and n is the number of events of P . The procedure "CheckFeasibility" loops over all the elements in map PCM , and maximum number of elements of each list under one key is N , so the maximum possible size of PCM is nN . The procedure "FindPaFast" only add an element into the list under a specific key in PCM , if PCM is implemented by some constant-time insertion data structure such as hash-map, this step will only take constant time then. Therefore, if we consider the two procedures together, the worst case time complexity of "Fast Pattern Matching" is $O(nN)$. Another important advantage of "Fast Pattern Matching" over the "Recursive Pattern Matching" introduced above is that it amortizes the work-load of pattern matching from the bar where E_n triggers into all the bars. In "Recursive Pattern Matching", the $O(2^N)$ recursive checking will be called on each event E_n being detected, which means all the time-consuming operations are performed after patterns already formed in the market. In real-time trading, since there might be a lot of other strategies try to take profit via the same patterns, the brute-force recursive procedure will cost a serious delay and make our strategies less competitive. In contrast, "Fast Pattern Matching" delegate the time-consuming work of checking feasibility to all the candle bars. And when event E_n is detected, the algorithm will return the pattern triggering signals in constant time, which minimize the delay between the real pattern triggering and MTPS responses.

5.1.4 Time-frame Alignment

In previous sections, the detailed process of MTPS Pattern Analysis was presented, and in this subsection, we will introduce how MTPS analyze multi-time-frame technical patterns. As mentioned in chapter , the capability of modeling multi-time-frame technical patterns is one of the most

distinguishable features of MTPS, and a multi-time-frame pattern can be either contains multi-time-frame search-bar events or contains events with different event-frames. To handle the two cases of multi-time-frame patterns, we employ a implementation trick "time-frame alignment" in MTPS.

"Time-frame alignment" refers to the process of how MTPS calculates the time difference between two characteristic candles on different time-frames. This trick is just to map all candles into the same time-frame. To achieve this, after the "event detection" step, MTPS will associate multiple indexes to each event generating "indexes-event" pairs. The indexes of an event are the time-index on the event-time-frame and the time-indexes on all the time-frames larger than the event-time-frame. For instance, a technical pattern P is to be detected by MTPS which is composed by events $[Event_1 Event_2]$ and time intervals $[T_1]$. Suppose the time-frame of $Event_1$ is daily, while time-frame of $Event_2$ is 60min, time-interval T_1 is "3 days", then P is a multi-time-frame technical pattern with two time-frames that are "daily" and "60min". In this case, on the detection of $Event_1$, MTPS will output a "daily-index, $Event_1$ " pair, while on the detection of $Event_2$, MTPS will output a "(daily-index, 60min-index), $Event_2$ " pair. Therefore, when calculating the time difference between $Event_1$ and $Event_2$, we can easily subtracting daily index of $Event_2$ from daily index of $Event_1$. Figure 5.3 shows how exactly time-frame alignment works for pattern P . In this figure, the coordinate of daily chart aligning with the coordinate of 60min chart. When performing the "pattern matching", MTPS Pattern Analyzer will map $Event_2$ into daily chart.

5.2 MTPS Market Data Pre-processing

The process of how to recognize technical patterns are thoroughly elaborated on the section above, and this process illustrates why Pattern Analyzer serves as the core components in MTPS and provides the major novelty of this paper. On the other hand, market data pre-processing also plays an essential role to drive the complete flow of strategy backtesting. In this section, we will present details of the MTPS market data pre-processing. The main task of market data pre-processing can also be divided into three steps that are "Data Sourcing", "Dynamic Candle Aggregation" and "Date Storing".



Figure 5.3: time-frame alignment of Pattern P

The "Data Sourcing" step is a process to import candle bars on 5min time-frame into memory from files downloaded of TradeStation. Ideal strategy backtesting systems will backtest technical strategies over tick data stream. Tick market data streams contain huge amount of data, which normally generate hundreds of data ticks per second per instrument, and hence can enhance the accuracy of backtesting. However, processing tick data streams is very computational intensive task and the prices to buy real-time tick market data stream from all data vendors are very high. Therefore, in our prototyping implementation of MTPS, we employ TradeStation as our market data source to simulate the behavior of real-time streaming data. By running Easy Language scripts in TradeStation, we are able to automate the market data downloading of TradeStation. Every time a new candle bar sent from exchanges to the TradeStation client charting window, this candle bar will be appended into a file in "date,time,open,high,low,close,volume" format. The DataUpdater of MTPS will then keep scanning the data files periodically and import the newly appended candle bar as long as there is any changes of the last edited time of the files. The frequency of the data importing is slightly smaller than 5 minutes which is the frequency of data downloading. Therefore, in every 5 minute, a new 5min bar will be imported via DataUpdater. Even though 5 minutes is still a too large delay for practical intra-day trading analysis, it will be easy to extend it to a commercial version to support time-frame less than 1 minute with a little code changes.

The "Dynamic Candle Aggregation" step will take use of the candle bars of 5min time-frame generated in "Data Sourcing" to aggregate candle bars of larger time-frames. Apart from 5min time-frame, 4 larger time-frames are supported in our MTPS implementation, that are 60min, daily, weekly, and monthly. It is also possible to download candle bars of all these time-frames from TradeStation, however, such directly downloading will cost 1 hour, 1 day, 1 week, and 1 month delay of patten recognition, which is intolerable for real-time trading. Therefore, MTPS aggregates candle bars of larger time-frames by the 5min time-frame bars dynamically. The algorithm of candle bar aggregation is that, suppose that the existing candle bar on large time-frame which is the aggregation base is $Bar_1(time_1, open_1, high_1, low_1, close_1, volume_1)$ and the newly update

This dynamic aggregation are applied on all the large time-frames. Take the 60min time-frame as an example, every time a new 5min candle bar is generated by "Data Sourcing", the DataUpdater will check if the last 60min candle bar has closed. If so, the 5min bar will be copied directly to

Algorithm 3 Candle Aggregation Algorithm

```
1: procedure CANDLE AGGREGATION
2:   Bar1  $\leftarrow$  (time1, open1, high1, low1, close1, volume1)aggregation base
3:   Bar2  $\leftarrow$  (time2, open2, high2, low2, close2, volume2)new generated candle bar
4:   Bar3  $\leftarrow$  (time3, open3, high3, low3, close3, volume3)aggregated result bar
5:   open3  $\leftarrow$  (open1)aggregating open
6:   if high1  $\geq$  high2 then
7:     high3  $\leftarrow$  (high1)aggregating high
8:   else
9:     high3  $\leftarrow$  (high2)aggregating high
10:  if low1  $\leq$  low2 then
11:    low3  $\leftarrow$  (low1)aggregating low
12:  else
13:    low3  $\leftarrow$  (low2)aggregating low
14:  close3  $\leftarrow$  (close2)aggregating close
15:  volume3  $\leftarrow$  (volume1 + volume2)aggregating volume
```

60min time-frame as a new 60min bar. If not, this 5min bar will be aggregated into the last 60min bar to update its open, high, low, close, volume. In such a aggregation manner, the candle bars of each time-frame forms a market data stream, and the generation or update of candle bars is an event to drive a move of the strategy execution.

The "Data Storing" step refers to the process of saving the candle bars and the values of technical indicators calculated on bars into database storage. In our implementation of MTPS, all the data are saved in SQLiteDB which is a light weight relational database. Normally, candle bars will be cached for a while in memory for indicators calculation and being written into database after being out-of-date. The database writing will be done asynchronously with the pattern analysis, so that the very time-consuming database writing won't become the bottleneck of the real-time trading. The market data will be first sent to a message queue and then being written into the database on other threads.

The MTPS market data pre-processing efficiently prepares the market data required for pattern analysis and strategy backtesting later. All the three steps of data pre-processing steps will generate fast moving market data streams to trigger strategy execution process, as well as keep data being stored in database.

5.3 Technical Indicators Implemented

Technical indicators are foundations of technical analysis, and we implement rich technical indicators in MTPS. After a candle stick bar being aggregated or updated, a group of technical indicators on this bar will be calculated and then be used for "Event Detection" in pattern analyzer. In our prototyping implementation of MTPS, all these indicators are only calculated based on default fixed parameter sets, while can be easily extended into user customized parameters. In chapter 3, we introduced the details of several most important technical indicators in MTPS. In this section, we will list all 12 kinds of indicators implemented in MTPS and give brief descriptions and default parameters of them. The first 9 kinds are the most commonly used indicators among technical traders and the last 3 kinds are designed by an experienced trader and uniquely implemented in MTPS.

1. Simple Moving Average

- Description: A simple moving average (SMA) is an arithmetic moving average of the closing price of the instrument over a time period n .
- Default Parameters: time period $n = 10$ as "Fast SMA" and $n = 50$ as "Slow SMA".

2. Exponential Moving Average

- Description: An exponential moving average (EMA) is a type of moving average over time period n that is similar to a simple moving average, except that more weight is given to the latest data[2].
- Default Parameters: time period $n = 10$ as "Fast EMA" and $n = 50$ as "Slow EMA".

3. Moving Average Convergence Divergence

- Description: Moving average convergence divergence (MACD) is a trend-following momentum indicator that shows the relationship between two moving averages of prices. The MACD is calculated by subtracting the m day exponential moving average (EMA) from the n day EMA [3].
- Default Parameters: time period $m = 26$ and $n = 12$.

4. Relative Strength Index

- Description: The relative strength index (RSI) is a momentum indicator that compares the magnitude of recent gains and losses over a specified time period n to measure speed and change of price movements of an instrument[4].
- Default Parameters: time period $n = 14$.

5. Stochastic Oscillator

- Description: The stochastic oscillator is a momentum indicator comparing the closing price of an instrument to the range of its prices over a certain period of time n . If we define a stochastic indicator as "Stochastic Fast", then its m bar moving average is called "Stochastic Slow".
- Default Parameters: time period $n = 14$ and smoothing factor $m = 3$.

6. Momentum

- Description: Momentum is the indicator measures the difference between the close price of current bar and the close price of n previous bar.
- Default Parameters: time period $n = 10$ as "Up Momentum" and $n = 50$ as "Down Momentum".

7. Rate Of Change

- Description: Rate Of Change (ROC) refers to the percentage of difference between current bar close price and the last bar close price.

8. On-Balance Volume

- Description: On-balance volume (OBV) is a momentum indicator that uses volume flow to predict changes in instrument price. The OBV can be calculated by adding the volume of current bar if current close price greater than last close price, vice versa[5].

9. Average True Range

- Description: The true range indicator is the greatest of the following: current high less the current low, the absolute value of the current high less the previous close and the absolute value of the current low less the previous close. The average true range is a moving average over time period n of the true ranges[6].
- Default Parameters: time period $n = 14$.

10. SMA Zone

- Description: SMA Zone (SMAZ) is one of the self-defined technical indicators in MTPS. SMAZ includes two SMA indicators of time periods of m and n . Suppose $m < n$ and denote that $SMA_m = x$, $SMA_n = y$ and the current bar close is c , then the value of SMAZ can be summarized as follows,

$$SMAZ = \begin{cases} 1 & c > x > y \\ 2 & x > c > y \\ 3 & x > y > c \\ 4 & c > y > x \\ 5 & y > c > x \\ 6 & y > x > c \end{cases}$$

- Default Parameters: time period $m = 10$ and $n = 50$.

11. Bar Shape

- Description: Bar Shape refers to a set of MTPS self-defined technical indicators to describe the shape feature of a candle stick bar including Up Tail(UT), Down Tail(DT), Body and Total Length(TL). For instance, UT is calculated by $(High - \max(Open, Close))$ and then being normalized by the mean and standard deviation of all up tails of the historical candle bars of corresponding instrument. Therefore, UT of the current bar is greater than 3, that means the up tail of this candle bar is greater than 3 standard deviations of all the historical candle bars.

12. Support and Resistance

- **Description:** Support and Resistance levels are MTPS self-defined indicators calculated based on the method mentioned on section. A support/resistance level is a price line that during a look-back period h , the price line is reacted at least n times with a react threshold r and is never broken. The price trends are highly probably kept or reversed when hitting these price levels.
- **Default Parameters:** look-back period $h = 250$, minimum react times $n = 2$ and react threshold r varies with different instruments.

5.4 MTPS Trading Components

Trading components of MTPS are "trading engine", "order management system", "strategy performance analyzer" and all the other components simulating trading procedure or participating into real-time trading. Trading components generally take the pattern signals from Pattern Analyzer and market candle bars as input, output trading orders to brokerage and strategy performance to users. In this section, we will briefly introduce that how the trading components of MTPS interact with the whole system and generate the outputs.

Trading engine serves as the controller of the whole flow of strategy execution. If users start to run strategies, trading engine will load the strategies from the saved files and assign all the patterns and events included in strategies to pattern analyzer. During the process of strategy execution, trading engine will control the pattern analyzer to recognize the active patterns based on the position holding status and listen to the pattern triggering signals. For example, if no position are hold by a strategy, trading engine will keep listening to the pattern defined in the entry rule of the strategy. On the other hand, if there are some positions hold by a strategy, trading engine will keep listening to the patterns defined in the exit rules of the strategy. As long as a pattern is listened to trigger, trading engine will send order placement or canceling commands to the order management system based on the trading rules defined in strategies. Meanwhile, trading engine will also collect the transactions of executed orders and take some further actions. For instance, if an order triggered by the entry rule of a strategy is successfully executed and the transaction returns to trading engine, a stop loss order based on the stop loss rule will be sent out simultaneously. The trading engine

will also send order execution transactions to the strategy performance analyzer to calculate and monitor strategy performance.

Order management system is controlled by the order commands from trading engine and handles all the order related tasks. Since users are able to select to do backtesting or real trading with money, the order management system is designed to run in two modes, the backtesting mode or the real-time trading mode. When running with the real trading mode, the order management system will forward orders to trading brokerage to get it traded. In our prototyping implementation of MTPS, we test this mode by implementing the connection to the Interactive Broker. Besides, the order execution transactions will be collected by order management system and returned to trading engine. Order management system can also retrieve the pending order lists from the brokerage and cancel orders if required by trading engine. When running with the backtesting mode, the order management system will just assume that all the market orders will be executed successfully on the current bar open price while limit orders or stop loss orders will be executed successfully on their desired prices. The order management system will then generate dummy transactions and return them to trading engine.

Strategy performance analyzer takes the order transactions as input and monitors the positions and analyze performance of all the running strategies. Before running strategies, performance analyzer will assign initial capitals and initial positions to them. The default initial position is zero and initial capital can be configured by users. During the strategy execution, performance analyzer keeps calculating and monitoring positions and total capital of strategies based on the order transactions from trading engine. Several popular strategy performance measurements mentioned in chapter 3, such as net profit, max-draw-down and sharp ratio, will also be calculated and reported to users. After the backtesting process terminated, a performance report, a transaction list and a curve of capital change will be printed to users. Figure 5.4 shows an example of the performance report. When running with real trading mode, performance analyzer will print all the statistics onto a strategy monitoring screen. Users may then fine-tune the parameters based on strategy performance.

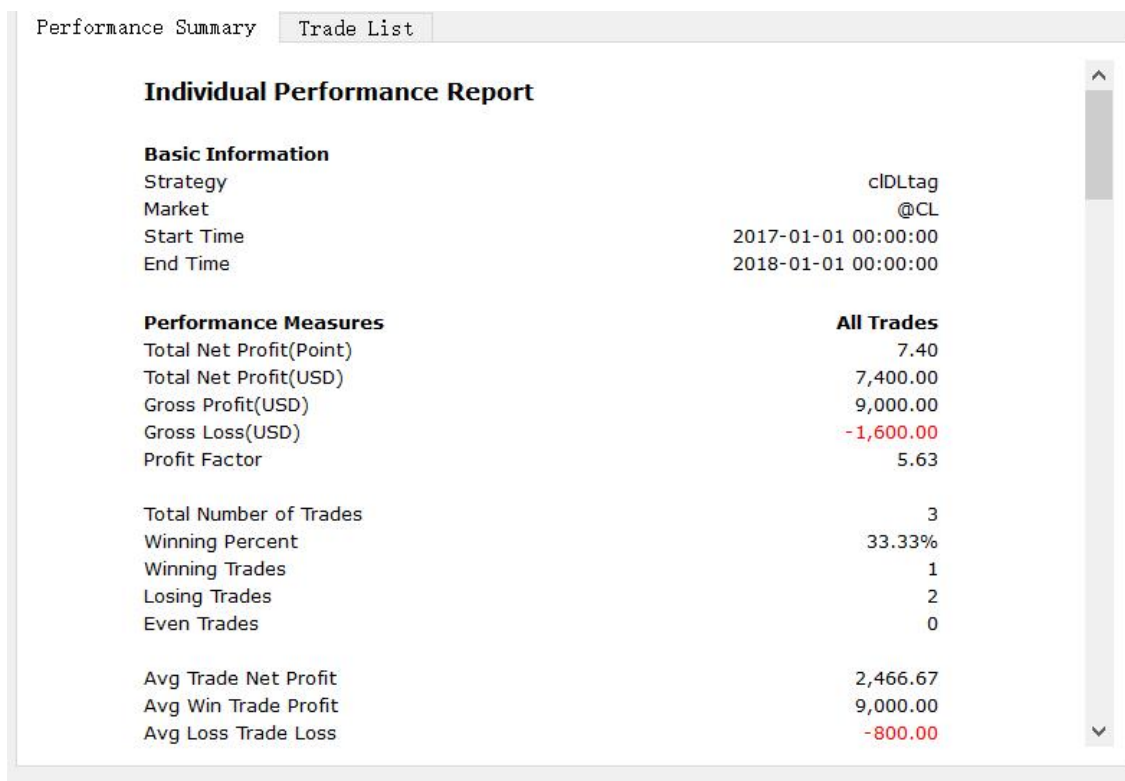


Figure 5.4: Performance Report GUI

5.5 Event-Driven Architecture of MTPS

Event-Driven Architecture (EDA) is a software design pattern to design software architecture for generating, monitoring, handling events reactively to accomplish some functionalities. Notice that the term "event" here is software engineering concept which is different from the technical event mentioned before. As mentioned in subsection 2.2.1, EDA is very powerful to reduce the coupling and increase responding speed of a software. This design pattern is very widely applied in developing graphical user interface and real-time data processing software. Strategy backtesting is a process to execute strategies over a real-time market data stream, therefore, many successful strategy backtesting systems such as TradeStation, NinjaTrader and MetaTrader are designed based on EDA. We have explored all the important software components of MTPS in the previous sections of this chapter, and in this section, we are going to illustrate how these components are organized in a fully asynchronous event-driven architecture.

In Event-Driven Programming, the concept "event" refers to messages that needed to be transmitted across different processes or threads. Software designed with EDA usually contains at least three kinds of components that are event producers, event consumers and event channels. Event producers generate events and publish them into the event channels. While event consumers subscribe events from event channels and handle the information within events. Event channels, in our implementation, are message queues which are able to guarantee that the dequeue order are exactly the same with the enqueue. Generally, messages or inputs that arrive in real-time or irregularly will be treated as events, therefore, in MTPS, market data update, order transaction feedback and trading signals are employed as events. Figure 5.5 shows the overview of how these events flow between components of MTPS. The details of the events are listed as follows,

- **Market Data Event** After candle bars are aggregated or updated, DataUpdater will produce market data events containing the corresponding bars and publish the events into the event channels, and the events will be consumed by Trading Engine and send to Pattern Analyzer to detect patterns. In our current implementation, DataUpdater will import data from data files periodically, and simulate a market data stream to produce market data events. It will be very easy to replace this data updater by a real-time data stream from data vendor.

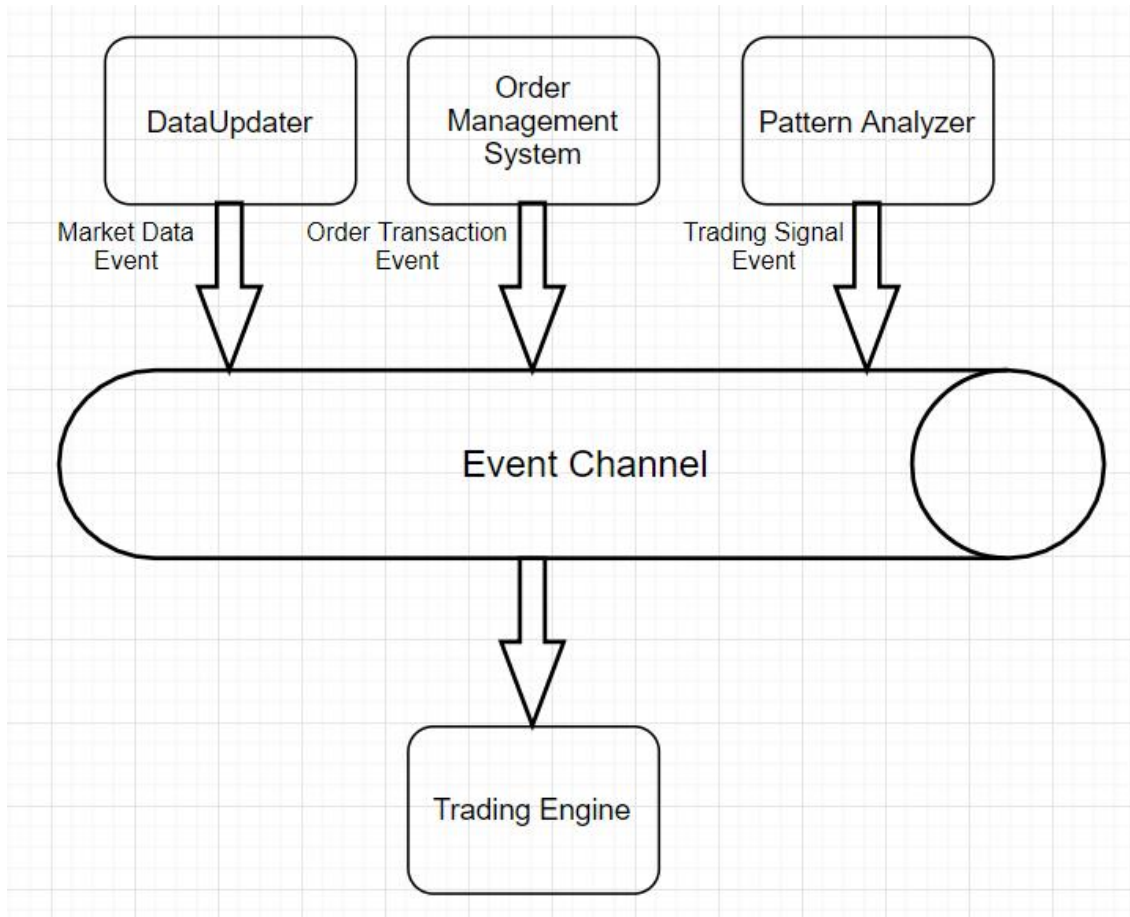


Figure 5.5: EDA in MTPS

- **Order Transaction Event:** The order management system will keep waiting to the feedback of order execution results from trading brokers or being simulated by itself and produce order transaction events containing the transactions of order execution into event channels. The order transaction events will be consumed by the Trading Engine to update strategy positions and adjust the active patterns to be detected in Pattern Analyzer.
- **Trading Signal Event:** When an active pattern is detected to trigger on a candle bar, Pattern Analyzer will produce the trading signal events containing the pattern id and the index of bar that pattern trigger on. These events will be consumed by Trading Engine to place corresponding orders to order management system.

CHAPTER 6

EXPERIMENTAL EVALUATION

In chapter 5, we describe the details on the implementation of MTPS. In this chapter, we will present and evaluate the power of MTPS by describing, simulating three technical strategies and show their performance. Additionally, our experiment will compare the performance of MTPS designed strategies with buy/sell and hold strategy to show their profitability.

The experiment of MTPS including the backtesting of three strategies. The first strategy, "Day-50 break pull-back" shows how users can design a profitable strategy by only taking a simplest technical indicator, 50 day simple moving average (SMA) into consideration. The second strategy is the classical technical strategy, "RSI Divergence". The third strategy, "Composite Long", is our self-defined technical strategy which consists of multiple technical indicators from different time-frames. It is designed specially to thoroughly test the capability of MTPS to describe and backtest complex multi-time-frame technical strategies.

6.1 Experimental Setup

Our experiments are performed on the implementation of MTPS described on chapter 5. The hardware used in experiments is just a PC with a 4 core Intel i7 CPU 3.6GHz and 16 GHz RAM. Three of most actively traded futures contracts data are selected for strategy simulations. Two commodity futures chosen are the Crude Oil contracts(symbol @CL) and the Gold contracts(symbol @GC) traded in New York Mercantile Exchange (NYMEX). The other one futures is the NASDAQ-100 contracts(symbol @NQ), which is an equity index futures traded in Chicago Mercantile Exchange. The reason why we choose these futures contracts as the experimental data is that the trading volume of these instruments are large enough so that it is almost impossible for them to be manipulated or to shake extremely. Also futures prices are affected much more slightly by sudden fundamental accidents than the prices of stocks or foreign exchanges. Therefore, the price movements of futures

are most likely to be dominated by technical trading laws and are the best data sets to backtest technical strategies. For each instrument, we assume the initial capital is 50,000 USD used to calculate the annualized return rate, which is more than sufficient for all the margin requirements and trading costs.

6.2 Experiment Results on "Day-50 break pull-back"

50 day moving average or say Day-50 is one of the most important technical indicators indicating long-term trends of the market. A very simple use case of Day-50 is just to buy an instrument when its daily bar close is greater than or equal to Day-50, and close the long positions when the daily bar close is less than Day-50. While in many cases, it is highly possible that the market price will go down a small distance after the first daily bar close above Day-50 until the market price is around the Day-50 again which is called the pull-backs of Day-50 break. Such pull-backs are normally regarded as last attempts of the short side of the market, and after pull-backs, the market may keep bullish for a long time. Figure 6.1 shows a perfect example of "Day-50 break pull-back", In this case, the purple line is the 50 day moving average, and the red circle shows that after the daily bar close above Day-50, in the next trading day, the daily low is around Day-50. Therefore if we place limit long orders at the price around Day-50 in the next trading day, we can take the maximum profit from this long up trend. Besides, in order to avoid the bias from one single fundamental conditions, we prefer to choose a backtesting period long enough with multiple up and down price movement trends. Therefore, the backtesting duration for @CL and @GC is from 1st January 2015 to 31st December 2017, and backtesting duration for @NQ is from 1st January 2007 to 31st December 2011.

Based on the technical analysis, we can design our "Day-50 break pull back" strategy via the MTPS designing framework as following.

Events:

- **Eve1:** Search-bar event, contains two "technical filters", that are,
 - Daily, Close Price, greater than or equal to, SMA_{50}



Figure 6.1: Day-50 break pull back

- Daily, Last Bar Close Price, less than, Last Bar SMA_{50}
- **Eve2:** Search-line event, that is
 - Daily, Open Price, greater than or equal to, SMA_{50}
 - Daily, Low Price, less than, $SMA_{50} + EventPara_1$
- **Eve3:** Displacement event. Denote the market price as P_m , the current entry price as P_e , and a parameter $EventPara_2$. Then it triggers if $P_m - P_e \geq EventPara_2$.

Patterns:

- **Pat1:** event list $[E_1, E_2]$ and time interval list T_1 .

Strategy

- **Entry Rule:** When Pattern $Pat1$ triggers and the strategy holds flat position, then buy 1 contract of futures.
- **Exit Rule:** When $Eve3$ triggers and the strategy holds long position, then sell 1 contract of futures at market price.
- **Stop Loss Rule:** When the market price go down from the entry price by L_s dollars and the strategy holds long position, sell 1 contract of futures at market price.

Instrument	@CL	@GC	@NQ
$EventPara_1$	0	2	6
$EventPara_2$	10	130	230
T_1	3	5	5
L_s	0.8	40	40

Table 6.1: Parameter Sets of Day-50 break pull-back

Instrument	@CL	@GC	@NQ
Strategy Profit	18800	15120	11200
Buy and Hold Profit	-18520	8750	4575
Strategy Maximum Draw-Down	7790	13250	4930
Buy and Hold Maximum Draw-Down	38590	26560	22445
Strategy Annualized Return Rate	11.23%	9.21%	6.92%
Total number of entry	17	9	13
Winning Entry Percentage	17.65%	33.33%	30.77%

Table 6.2: Performance of Strategy "Day-50 break pull-back"

From the definition of "Day-50 break pull-back", it is easily to figure out that the strategy contains 4 parameters need to be customized, i.e. $EventPara_1$, $EventPara_2$, T_1 , and L_s . These parameters need to be carefully tuned based on the historical data of each instrument to maximize the profit and minimize the risk. In experiments in this chapter, we will use parameters tuned manually by an experienced trader.

We then backtest the "Day-50 break pull-back" strategy on the 3 instruments with parameter sets listed in Table 6.1 and present the cumulative profit and maximum draw-down. We also compare the performance of the strategy with the "Buy and Hold" strategy which represents the general performance of the market. In Table 6.2, we present the performance of "Day-50 break pull-back" on the 3 instruments backtesting over 3 years of market data. It is quite obvious that for all the 3 instruments, the profits of "Day-50 break pull-back" are much higher than "Buy and Hold", while the maximum draw-downs are much smaller, which means by applying this strategy, traders can earn much more money with a much lower risk. Figure 6.2, figure 6.3 and figure 6.4 show the curves of profit and loss over time of strategy "Day-50 break pull-back" and "Buy and hold" of the three instruments. From these figures, it is easily to conclude that strategy "Day-50 break pull-back" can gain better and more stable profit than "Buy and hold" at most of the time during backtesting.

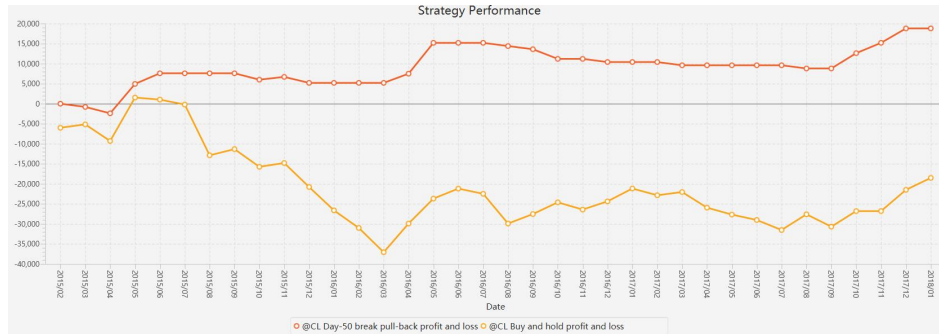


Figure 6.2: profit of "Day-50 break pull-back" vs. "Buy and hold" on Crude Oil



Figure 6.3: profit of "Day-50 break pull-back" vs. "Buy and hold" on Gold

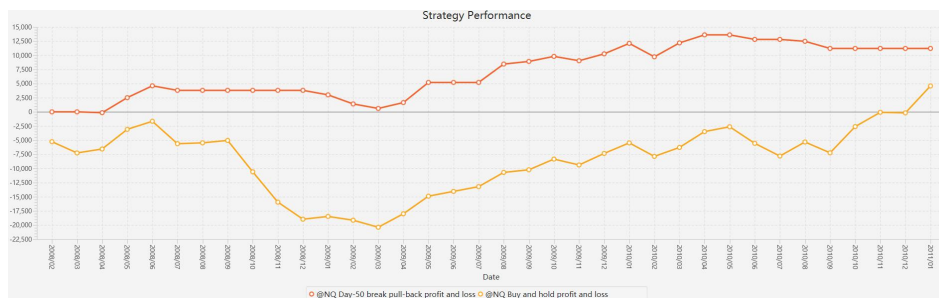


Figure 6.4: profit of "Day-50 break pull-back" vs. "Buy and hold" on NASDAQ

6.3 Experiments Results on "Hourly RSI Divergence"

Momentum indicator divergence is a kind of most famous and commonly used technical patterns, which have been discussed detailedly in Chapter 3 and Chapter 4. In this experiment, we are going to present how to design a strategy based on pattern "Hourly RSI Divergence" which is a specific example of momentum indicator divergence and show the simulation results of this strategy.

Events:

- **Eve1:** Search-bar event, with technical filter
 - Hourly, RSI, less than, $EvePara_1$
- **Eve2:** Search-bar event, with technical filter
 - Hourly, RSI, greater than or equal to, $EvePara_2$
- **Eve3:** Search-bar event, with two technical filters
 - Hourly, Low Price, less than, the lowest price in last $EventPara_3$ hours
 - Hourly, RSI, greater than or equal to, the lowest hourly RSI in last $EventPara_3$ hours
- **Eve4:** Search-bar event, with technical filter,
 - Hourly, RSI, less than, $EvePara_4$
- **Eve5:** Displacement event. Denote the market price as P_m , the current entry price as P_e , and a parameter $EventPara_5$. Then it triggers if $P_m - P_e \geq EventPara_5$.

Patterns:

- **Pat1:** Event list $[Eve1, Eve2, Eve3, Eve4, Eve2]$, time interval list $[T_1hours, T_2hours, 0hours, T_3hours]$.

Strategy

- **Entry Rule:** When Pattern $Pat1$ triggers, buy 1 contract of futures.



Figure 6.5: Typical "Hourly RSI Divergence"

- **Exit Rule:** When *Eve5* triggers, sell 1 contract of futures at market price.
- **Stop Loss Rule:** When the market price go down from the current entry price by L_s dollars, sell 1 contract of futures at market price.

The *Pat1* defined above describes the typical shape of "Hourly RSI Divergence". *Eve1*, *Eve4* describe bottoms of market, *Eve2* describes a recover from bottoms, while *Eve3* describes divergence of RSI and price. Therefore, the whole pattern represents a sequence of market price movements, that is the market first goes sharply down to a bottom, then recovers, then goes down to a second bottom that is lower than the first bottom while the RSI of second bottom is higher than the first bottom, and eventually, recovers again. Such a recover indicates the end of this "Double bottom" wave and a beginning of a up trend. Figure 6.5 gives an very typical example of "Hourly RSI Divergence" on Crude Oil at 20th Oct 2017 found by MTPS pattern analyzer. Apparently, there are 9 parameters in this "Hourly RSI Divergence", and also we adopt one set of parameters

Instrument	@CL	@GC	@NQ
$EventPara_1$	27	27	26
$EventPara_2$	30	30	30
$EventPara_3$	24	24	24
$EventPara_4$	27	27	28
$EventPara_5$	2.3	15	90
T_1	12	24	24
T_2	12	24	24
T_3	12	24	24
L_s	0.7	4	35

Table 6.3: Parameter Sets of Hourly RSI Divergence

Instrument	@CL	@GC	@NQ
Strategy Profit	31860	12510	12000
Buy and Hold Profit	-18520	8750	4575
Strategy Maximum Draw-Down	7740	3490	6800
Buy and Hold Maximum Draw-Down	38590	26560	22445
Strategy Annualized Return Rate	17.86%	7.73%	7.43%
Total number of entry	53	58	40
Winning Entry Percentage	43.4%	34.48%	40%

Table 6.4: Performance of Strategy "Hourly RSI Divergence"

roughly tuned by a trader on each of the instrument and present them in Table 6.3.

Table 6.4, Figure 6.6, Figure 6.7 and Figure 6.8 show the backtesting results of "Hourly RSI Divergence" on the 3 instruments. We can easily conclude that "Hourly RSI Divergence" strategy can significantly beat the "Buy and Hold" on both profit and risk perspectives over Crude Oil and NASDAQ. While backtesting over Gold, the performance of two strategies are similar on profit, but "Hourly RSI Divergence" still suffers a much smaller gross loss during the backtesting period.

6.4 Experiment Results on "Down Trend Exhausted"

In the experiments on last two sections, we describe and backtest two strategies based on universally known technical patterns "Day-50 break pull-back" and "Hourly RSI Divergence" on MTPS. Our results prove that MTPS is able to model some delicate and classical technical patterns and capture

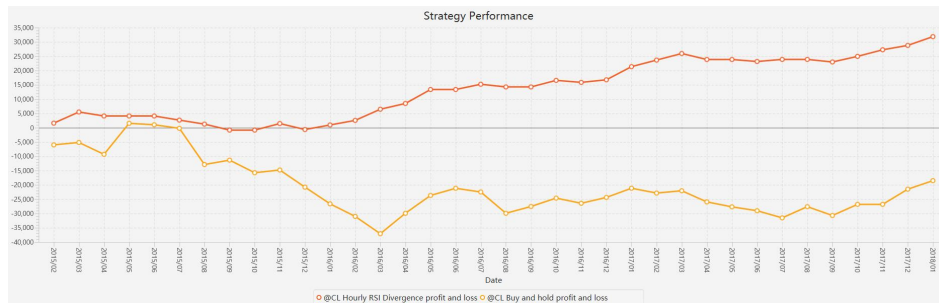


Figure 6.6: profit of "Hourly RSI Divergence" vs. "Buy and hold" on Crude Oil

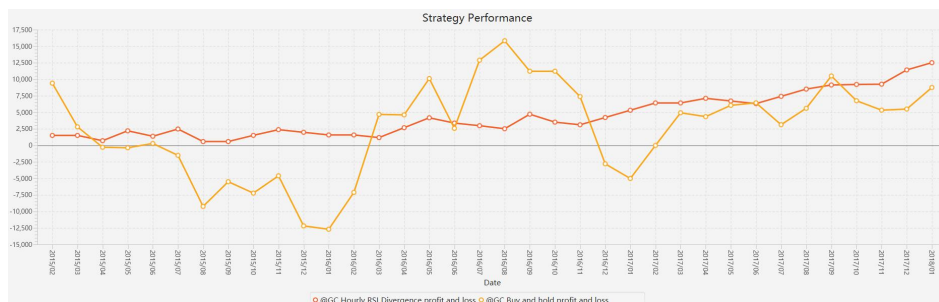


Figure 6.7: profit of "Hourly RSI Divergence" vs. "Buy and hold" on Gold

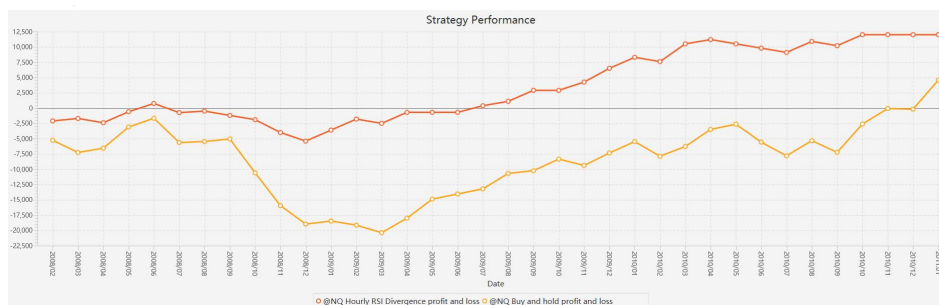


Figure 6.8: profit of "Hourly RSI Divergence" vs. "Buy and hold" on NASDAQ

them in market data streams. However, these two technical patterns are still single time-frame patterns. To fully test the capability of modeling multi-time-frame technical patterns, in this section, we will backtest a strategy whose entries are triggered by a self-defined multi-time-frame technical pattern, "Down Trend Exhausted". This pattern is designed specifically for Crude Oil, so we only show the backtesting results on Crude Oil.

When designing "Down Trend Exhausted", our goal is to find a long entry signal triggers by several different indicators from different time-frame. Under these circumstances, we come out the basic idea of the pattern that is after a daily bar RSI being over-sold, at the next trading day, the market price also drops quickly such that the hourly RSI also over-sold, then market price hits an important Resistance level, and eventually the market recovers quickly to confirm the exhaustion of short momentum indicating a beginning of a long trend. Based on such idea, we can define this "Down Trend Exhausted" in the MTPS strategy description system as follows,

Events:

- **Eve1:** Search-bar event, with technical filter,
 - Daily, RSI, less than, $EventPara_1$
- **Eve2:** Search-bar event, with technical filter,
 - Hourly, RSI, less than, $EventPara_2$
- **Eve3:** Search-line event, with lines defined as
 - Daily, Support Levels, Distance, greater than or equal to, $EventPara_4$

Here the technical filter $Distance \geq 1$ means for support line Sup' that there is no any other support lines Sup' , such that $Sup - Sup' \leq EventPara_3$. And then the technical filter is as follows,

- Hourly, Market Price, less than, $Sup + EventPara_4$.
- **Eve4:** search-bar event, with technical filter,

- Hourly, SMA_{10} , greater than or equal to, SMA_{50}
- Hourly, last bar SMA_{10} , less than , last bar SMA_{50}
- **Eve5:** Displacement event. Denote the market price as P_m , the current entry price as P_e , and a parameter $EventPara_5$.Then it triggers if $P_m - P_e \geq EventPara_5$.

Patterns:

- **Pat1:** Events list [$Eve1, Eve2, Eve3, Eve4$], time interval list [$1Days, T_1hours, T_2hours$].

Strategy

- **Entry Rule:** When Pattern $Pat1$ triggers, buy 1 contract of futures.
- **Exit Rule:** When $Eve5$ triggers, sell 1 contract of futures at market price.
- **Stop Loss Rule:** When the market price go down from the current entry price by L_s dollars, sell 1 contract of futures at market price.

The definition above describes a straightforward idea of technical pattern: the Daily RSI($Eve1$) and hourly RSI over-sold($Eve2$) means that the short-sellers in the market have already accumulate a lot of floating profit and have strong will to cover or go long to realize their profit. On the other hand, price moving approaching the Resistance level($Eve3$) and price go up soon($Eve4$) means the long-buyers start to entry into the market. Therefore, by combining these four events sequentially, a long entry signal is generated. In Figure 6.9, an example of pattern "Short Trend exhausted" detected by MTPS is presented.

There are also 8 parameters of "Short Trend Exhausted" strategy tuned by an experienced futures trader. The details of these parameters are listed in Table 6.10. After backtesting the "Short Trend Exhausted" with this parameter set on Crude Oil futures data from 2007 to 2018, we got the results on Table 6.11 and Figure 6.12. Table 6.11 shows that even though from 2007 to 2018 the price of crude oil price drops from 130 dollar/barrel to less than 60 dollar/barrel, the strategy "Short Trend Exhausted" can still earn a relatively attractive annualized return rate of 4.51%, with a very

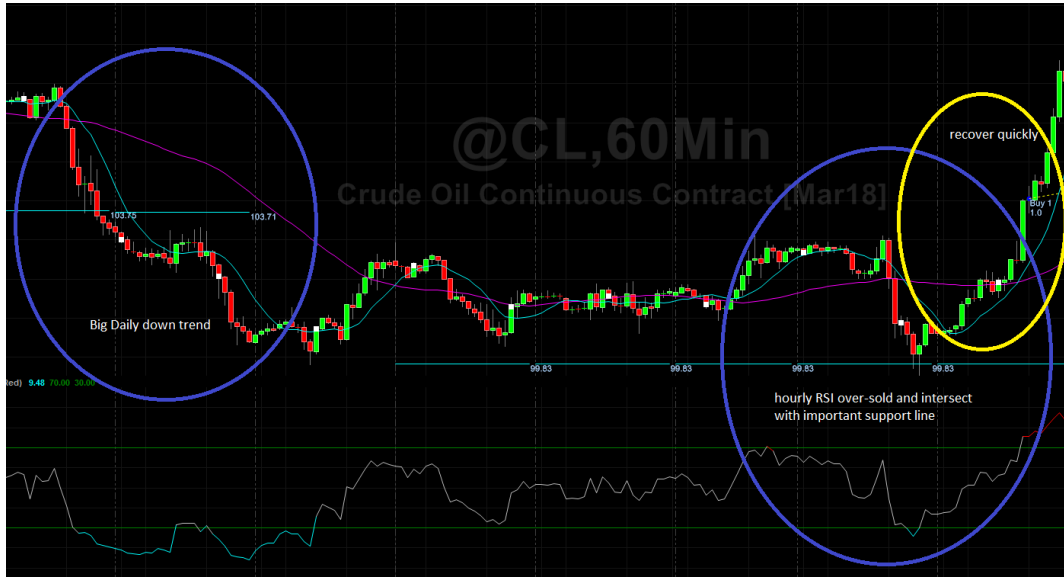


Figure 6.9: example of "Down Trend Exhausted"

Instrument	$EventPara_1$	$EventPara_2$	$EventPara_3$	$EventPara_4$	$EventPara_5$	T_1	T_2	L_s
@CL	28	30	1	0.2	8.5	3	24	2.6

Table 6.5: Parameter Sets of "Short Trend Exhausted"

small maximum draw-down. Figure 6.11 shows that "Short Trend Exhausted" can stably generate profit regardless of the great market volatility in last ten years. The results verify the capability of MTPS to detect complex cross time-frame technical patterns and show how traders can build their own strategies on technical patterns.

Instrument	@CL
Strategy Profit	32100
Buy and Hold Profit	-70620
Strategy Maximum Draw-Down	7800
Buy and Hold Maximum Draw-Down	175920
Strategy Annualized Return Rate	4.51%
Total number of entry	9
Winning Entry Percentage	56.56%

Figure 6.10: Performance of Strategy "Short Trend Exhausted"

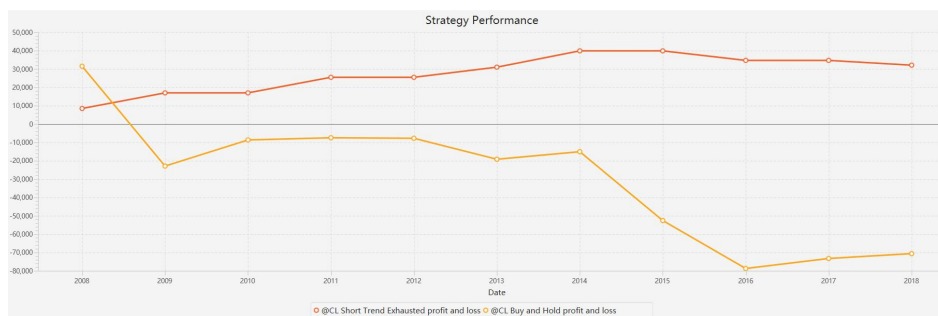


Figure 6.11: profit of "Short Trend Exhausted" vs. "Buy and hold" on Crude Oil

CHAPTER 7

CONCLUSION AND FUTURE WORKS

7.1 Conclusion

In this thesis, we propose the Multi-Time-Frame Technical Pattern and Strategy Backtesting System(MTPS) to address the problem about empowering technical traders to design advanced strategies via pure GUI and backtest strategies on market data streams. The system will be easily modified into a fully automated algorithmic trading system by connecting to real market data streams and real trading brokers.

In Chapter 3, knowledge of technical analysis included in MTPS are presented. We introduce several most commonly technical indicators implemented in MTPS and several examples of technical patterns that is able to be modeled in MTPS. We also introduce concept of multi-time-frame technical analysis and its power in trading. Last but not the least, several measurements for evaluating the performance of strategies.

In Chapter 4, the most important contribution of MTPS, the 3-layer hierarchical strategy description system and its corresponding GUI are illustrated. By taking use of the discipline that many advanced technical patterns can be decomposed into sequences of characteristic candle bars, MTPS promote users to describe technical patterns in two steps, describing technical events which are characteristic candle bars and then describing sequences of events which are technical patterns. Compared with all the state-of-the-art backtesting systems, this method of describing patterns keeps the user interactions purely via GUI while maximumly maintaining the flexibility of patterns. In addition, we also explore how MTPS allow users to describe multi-time-frame technical patterns.

In Chapter 5, we elaborate the design and implementation of MTPS. The whole architecture of MTPS is presented and several most important components are detailedly described. Pattern analyzer is the core component of MTPS, it is in charge of indexing candle bars, detecting technical events, matching patterns and eventually generating trading signals. We also propose a "Fast

Pattern Matching” algorithm which will cost a much smaller delay compared with the brute-force algorithm. In addition, the procedure of market data pre-processing is also shown which includes ”Data Scanning”, ”Candle Aggregation” and ”Data Storing”. Moreover, all the trading related components such as ”trading engine”, ”order management system” and ”strategy performance analyzer” are introduced. Eventually, we also describe how all the components of MTPS are organized within a event-driven architecture.

In Chapter 6, several experiments are conducted which mainly focus on verifying the power of description of advanced strategies of MTPS. Three advanced technical strategies including multiple technical indicators from multiple time-frames are described via MTPS GUI. We then backtest them on MTPS, plot the performance of them and show their profitability over buy-and-hold strategies.

7.2 Future Works

There are several future works might be conducted to enrich the functionalities of MTPS.

- Describing more advanced technical patterns: The main contribution of MTPS based on assumption that many technical patterns can be decomposed into a sequence of technical events. However, there are still a large amount of technical patterns have more complex structures. We may possibly extend MTPS to describe patterns formed by tree structured technical events or directed acyclic graph (DAG) structured technical events. With these two typological structures, users are able to describe almost all technical charting patterns via MTPS GUI.
- Optimizing parameters in patterns and strategies: When describing strategies in MTPS, users shall specify many parameters in GUI based on their own trading experiences. In the future work, We may employ optimization algorithms such as genetic algorithm and particle swarm optimization algorithms to optimize these parameters on historical market data. The performance of strategies may server as the optimization objectives.

- Including risk control functions: Risk control is an important concern when designing strategies. In our current work, we have included the stop loss trading rule as a basic risk control tool. In the future, our strategy backtesting system may allow users to control risk more actively by specifying how to allocate capital and restrict positions dynamically.

REFERENCES

- [1] <https://tradingsim.com/blog/trading-multiple-time-frames/>.
- [2] <https://www.investopedia.com/terms/e/ema.asp>.
- [3] <https://www.investopedia.com/terms/m/macd.asp>.
- [4] <https://www.investopedia.com/terms/r/rsi.asp>.
- [5] <https://www.investopedia.com/terms/o/onbalancevolume.asp>.
- [6] <https://www.investopedia.com/terms/a/atr.asp>.
- [7] Influxdb is 27x faster vs mongodb for time-series workloads. <https://www.influxdata.com/blog/influxdb-is-27x-faster-vs-mongodb-for-time-series-workloads/>.
- [8] Influxdb tops cassandra in time series data and metrics benchmark. <https://www.influxdata.com/blog/influxdb-vs-cassandra-time-series/>.
- [9] Mongodb delivers on tick database requirements. <https://www.mongodb.com/post/45116404296/how-banks-use-mongodb-as-a-tick-database>.
- [10] Time series database (tsdb) explained. <https://www.influxdata.com/time-series-database/>.
- [11] Jagrati Agrawal, Yanlei Diao, Daniel Gyllstrom, and Neil Immerman. Efficient pattern matching over event streams. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 147–160. ACM, 2008.
- [12] Aurélien Alfonsi and Alexander Schied. Optimal trade execution and absence of price manipulations in limit order book models. *SIAM Journal on Financial Mathematics*, 1(1):490–522, 2010.

- [13] Franklin Allen and Risto Karjalainen. Using genetic algorithms to find technical trading rules. *Journal of financial Economics*, 51(2):245–271, 1999.
- [14] Salim K Amirdache. Tss: A trading strategy system. 2010.
- [15] Alexander Artikis, Marek Sergot, and Georgios Paliouras. An event calculus for event recognition. *IEEE Transactions on Knowledge and Data Engineering*, 27(4):895–908, 2015.
- [16] Marco Avellaneda and Jeong-Hyun Lee. Statistical arbitrage in the us equities market. *Quantitative Finance*, 10(7):761–782, 2010.
- [17] Bruno Biais. High frequency trading. 2011.
- [18] Jkdrzej Białkowski, Serge Darolles, and Gaelle Le Fol. Improving vwap strategies: A dynamic volume approach. *Journal of Banking & Finance*, 32(9):1709–1722, 2008.
- [19] William Brock, Josef Lakonishok, and Blake LeBaron. Simple technical trading rules and the stochastic properties of stock returns. *The Journal of finance*, 47(5):1731–1764, 1992.
- [20] João Caldeira and Guilherme V Moura. Selection of a portfolio of pairs based on cointegration: A statistical arbitrage strategy. 2013.
- [21] Badrish Chandramouli, Jonathan Goldstein, and David Maier. High-performance dynamic pattern matching over disordered streams. *Proceedings of the VLDB Endowment*, 3(1-2):220–231, 2010.
- [22] Pei-Chann Chang, Chin-Yuan Fan, and Chen-Hao Liu. Integrating a piecewise linear representation method and a neural network model for stock trading points prediction. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 39(1):80–92, 2009.
- [23] Riccardo Curcio, Charles Goodhart, Dominique Guillaume, and Richard Payne. Do technical trading rules generate profits? conclusions from the intra-day foreign exchange market. *International Journal of Finance & Economics*, 2(4):267–280, 1997.

- [24] Frank Dabek, Nickolai Zeldovich, Frans Kaashoek, David Mazières, and Robert Morris. Event-driven programming for robust software. In *Proceedings of the 10th Workshop on ACM SIGOPS European Workshop*, EW 10, pages 186–189, New York, NY, USA, 2002. ACM.
- [25] Joey Fundora. <https://www.investopedia.com/articles/trading/07/timeframes.asp>, 2018.
- [26] G Pui Cheong Fung, J Xu Yu, and Wai Lam. Stock prediction: Integrating text mining approach using real-time news. In *Computational Intelligence for Financial Engineering, 2003. Proceedings. 2003 IEEE International Conference on*, pages 395–402. IEEE, 2003.
- [27] Nguyen Hoang Hung and Yang Zhaojun. Profitability of applying simple moving average trading rules for the vietnamese stock market. *J. Bus. Manag*, 2(3):22–31, 2013.
- [28] Masashi Ieda. A dynamic optimal execution strategy under stochastic price recovery. *International Journal of Financial Engineering*, 2(04):1550025, 2015.
- [29] Ieabeling Kaastra and Milton S Boyd. Forecasting futures trading volume using neural networks. *Journal of Futures Markets*, 15(8):953–970, 1995.
- [30] Michael Kearns, Alex Kulesza, and Yuriy Nevmyvaka. Empirical limitations on high-frequency trading profitability. *The Journal of Trading*, 5(4):50–62, 2010.
- [31] Thomas Kilgallen. Testing the simple moving average across commodities, global stock indices, and currencies. *The Journal of Wealth Management*, 15(1):82–100, 2012.
- [32] Richard Krivo. https://www.dailyfx.com/forex/education/trading_tips/post_of_the_day/2011/11/23/Multiple_Time_Frame_Analysis.html, 2011.
- [33] Ren Jie Kuo, CH Chen, and YC Hwang. An intelligent stock trading decision support system through integration of genetic algorithm based fuzzy neural network and artificial neural network. *Fuzzy sets and systems*, 118(1):21–45, 2001.

- [34] William Leigh, Naval Modani, Russell Purvis, and Tom Roberts. Stock market trading rule discovery using technical charting heuristics. *Expert Systems with Applications*, 23(2):155–159, 2002.
- [35] William Leigh, Russell Purvis, and James M Ragusa. Forecasting the nyse composite index with technical analysis, pattern recognizer, neural network, and genetic algorithm: a case study in romantic decision support. *Decision support systems*, 32(4):361–377, 2002.
- [36] Camillo Lento and Nikola Gradojevic. The profitability of technical trading rules: A combined signal approach. *Journal of Applied Business Research (JABR)*, 23(1), 2011.
- [37] Xuan Liu and Liu Pan. Prediction of the moving direction of google inc. stock price using support vector classification and regression. *Asian Journal of Finance & Accounting*, 6(1):323–336, 2014.
- [38] Andrew W Lo, Harry Mamaysky, and Jiang Wang. Foundations of technical analysis: Computational algorithms, statistical inference, and empirical implementation. *The journal of finance*, 55(4):1705–1765, 2000.
- [39] Junshui Ma and Simon Perkins. Time-series novelty detection using one-class support vector machines. In *Neural Networks, 2003. Proceedings of the International Joint Conference on*, volume 3, pages 1741–1745. IEEE, 2003.
- [40] Kazuhiro Matsui and Haruo Sato. Neighborhood evaluation in acquiring stock trading strategy using genetic algorithms. In *Soft Computing and Pattern Recognition (SoCPaR), 2010 International Conference of*, pages 369–372. IEEE, 2010.
- [41] Giovanni Montana, Kostas Triantafyllopoulos, and Theodoros Tsagaris. Flexible least squares for temporal data mining and statistical arbitrage. *Expert Systems with Applications*, 36(2):2819–2830, 2009.
- [42] Arman Khadjeh Nassirtoussi, Saeed Aghabozorgi, Teh Ying Wah, and David Chek Ling Ngo. Text mining for market prediction: A systematic review. *Expert Systems with Applications*, 41(16):7653–7670, 2014.

- [43] Yuriy Nevmyvaka, Yi Feng, and Michael Kearns. Reinforcement learning for optimized trade execution. In *Proceedings of the 23rd international conference on Machine learning*, pages 673–680. ACM, 2006.
- [44] Matt Piotrowski. <http://energyfuse.org/oil-market-volatility-and-the-quants-w> 2015.
- [45] Jean-Yves Potvin, Patrick Soriano, and Maxime Vallée. Generating trading rules on the stock markets with genetic programming. *Computers & Operations Research*, 31(7):1033–1047, 2004.
- [46] Ansar Rafique. Evaluating nosql technologies for historical financial data, 2013.
- [47] Reetinder Sidhu and Viktor K Prasanna. Fast regular expression matching using fpgas. In *Field-Programmable Custom Computing Machines, 2001. FCCM’01. The 9th Annual IEEE Symposium on*, pages 227–238. IEEE, 2001.
- [48] T Smith and M Waterman. ^aidentification of common molecular subsequences. ^oj. *Molecular Biology*, 147:195–197, 1981.
- [49] George Washington. <http://www.zerohedge.com/contributed/2012-17-26/84-all-stock-trades-are-high-frequency-computers-\%E2\%80\%A6-only-16-are-done-human-tra>, 2012.
- [50] Wing-Keung Wong, Meher Manzur, and Boon-Kiat Chew. How rewarding is technical analysis? evidence from singapore stock market. *Applied Financial Economics*, 13(7):543–551, 2003.
- [51] Haiqin Yang, Laiwan Chan, and Irwin King. Support vector machine regression for volatile stock market prediction. *Intelligent Data Engineering and Automated Learning IDEAL 2002*, pages 143–152, 2002.